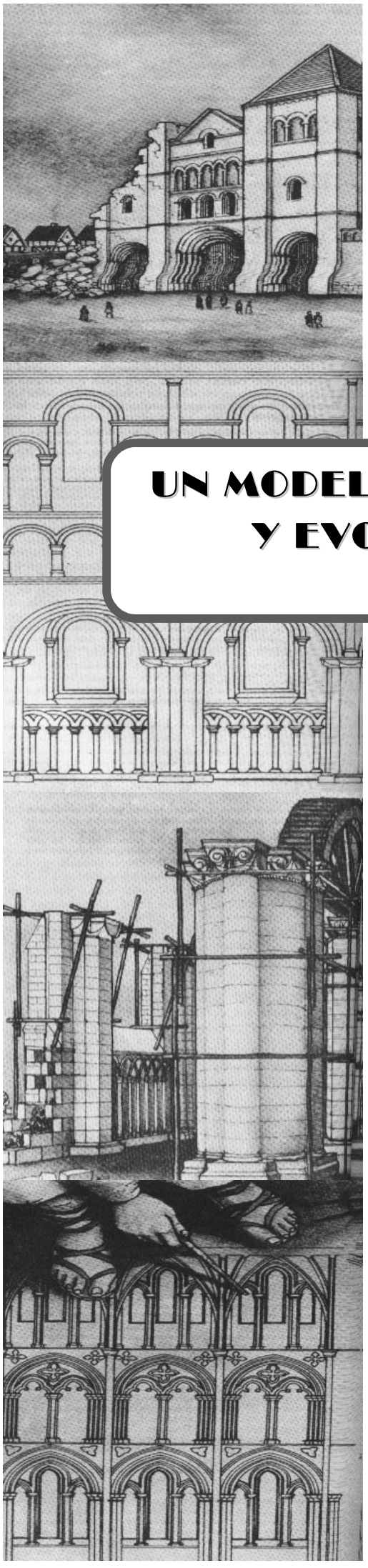


UN MODELO DE ADAPTACIÓN INTEGRAL Y EVOLUTIVO PARA SISTEMAS HIPERMEDIA

El Sistema de Aprendizaje de SEM-HP

NURIA MEDINA MEDINA

TESIS DOCTORAL

GRANADA - 2004

Universidad de Granada
Escuela Técnica Superior de Ingeniería Informática



Departamento de Lenguajes y Sistemas Informáticos

Un Modelo de Adaptación Integral y Evolutivo para Sistemas Hipermedia. El Sistema de Aprendizaje del Modelo SEM-HP

TESIS DOCTORAL

POR

NURIA MEDINA MEDINA

Memoria para optar al grado de Doctor en Informática

Codirigida por los profesores:

Dr. José Parets Llorca
Profesor Titular de Universidad
Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Dr. Lina García Cabrera
Profesor Titular de Universidad
Dpto. Informática
Universidad de Jaén

La memoria titulada ”Un Modelo de Adaptación Integral y Evolutivo para Sistemas Hipermedia. El Sistema de Aprendizaje de SEM-HP”, que presenta Dña. Nuria Medina Medina para optar al grado de Doctor, ha sido realizada en el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada bajo la dirección de D. José Parets Llorca y Dña. Lina Guadalupe García Cabrera.

Granada, Septiembre de 2004

La Doctoranda	El Director	La Directora
Nuria Medina Medina	José Parets Llorca	Lina García Cabrera

A mis padres

A Fernando

A veces uno no sabe cuánto ni cómo puede cambiar su vida de un día para otro. Una oportunidad despierta en nosotros la necesidad de hacer algo que nunca antes habíamos considerado. Todo comienza con un paso hacia adelante, después uno detrás de otro... Lo difícil llega cuando el camino es más largo de lo que pensamos, quizá porque ni siquiera nos habíamos detenido a hacerlo.

En ese momento giramos la vista alrededor y vemos que estamos en mitad de ningún sitio, a media distancia entre nuestro origen y nuestro destino. ¿Qué podemos hacer, sino seguir? Un exagerado miedo nos sobreviene, parece que el camino se estira, las distancias se alargan, ¿acabará esto alguna vez?, nos preguntamos.

Espero que la respuesta sea sí, porque estoy escribiendo estas líneas antes de haber llegado completamente al final, pero queda "poco", desde aquí ya se ve... Lo realmente divertido es que es el final de una carrera, pero seguro que no será la única, habrá muchas otras. Sin embargo, es imprescindible un descanso, la recompensa moral de haber superado una meta, en otro caso no podría afrontar un nuevo reto, y mucho menos hacerlo cuerda.

En cualquier caso, me alegro de ese día en que pensé: ¿Por qué no? ¡Vamos a intentarlo! Me alegro de haber podido emprender este viaje, de haber tenido la oportunidad...

He tenido que dejar algunas cosas en el camino, pero no llego con los brazos vacíos. Son muchas las cosas que he ganado. Algunas no podré apreciarlas hasta dentro de algún tiempo, con el título de doctora en la mano. Otras, hace ya mucho que conozco lo valiosas que son, precisamente gracias a ellas hoy llego hasta aquí, con una sonrisa.

Agradecimientos

Gracias a todos aquellos que por devoción u obligación lean esta memoria de tesis.

Gracias a mis directores por trabajar conmigo en agosto. Gracias a Pepe por su increíble capacidad para ver el conjunto de las cosas. Gracias a Lina por sus meticulosas correcciones y por ese toque artístico tan suyo.

Gracias a mis compañeros por ofrecerse a revisar mi trabajo, por sus consejos, su apoyo y su cariño. Especialmente a mis compañeros de despacho en Informática y en Empresariales: Mavi y Miguel Hornos. Y a mis vecinas de pasillo: María José y Patricia. También a Pepe Samos por sus sugerencias de última hora.

Gracias, asimismo, al resto de miembros de mi grupo de investigación GEDES: Ana, Miguel Gea, Francis, José Luis Garrido, Marisa, Ramón, Marcelino, Jesús Torres, José Luis Isla y Nicolás. Y, en general, a todos mis compañeros de departamento por el cariño con el que me tratan.

Gracias a mi familia, a mis amigos y a todos los que me quieren, por estar ahí.

Gracias a mi hermano por echarle siempre gasolina al coche, y luego irse andando.

Gracias a mi padre por apoyarme desde el umbral de la puerta, sin necesidad de palabras.

Gracias a mi madre por concederme mi tiempo todo para mí, por nunca pedirme ayuda. Por seguir tratándome como si fuese una niña.

Gracias a Gofy por hacerme feliz sin necesidad de hacer nada.

Gracias a mi “amigo” Fernando por permitirme mezclar tantas veces trabajo y placer. Por escuchar con infinita paciencia mis preocupaciones y consolarme cada vez como si fuese la primera. Muchas gracias por hacerme siempre su centro de atención, olvidando sus propios problemas ¡Ánimo Fernando!

Índice General

Prefacio

Punto de partida

El modelo SEM-HP fue propuesto por la doctora García-Cabrera en su tesis doctoral [García, 01c] hace ya tres años. Se trata de un modelo Sistémico, Evolutivo y SEMántico para el diseño de sistemas HiPermedia que surge con una motivación muy clara: definir y separar las fases que forman el proceso de desarrollo de Sistemas Hipermedia y asistir a sus autores, durante este proceso, de manera evolutiva.

El modelo final cumplía sobradamente sus objetivos, aplicando la evolución al desarrollo e implantación de los sistemas hipermedia. Debido a su propósito, el modelo se ocupaba especialmente de las tareas de autor, proponiendo como trabajo futuro varios aspectos relacionados con la interacción de los usuarios en el sistema.

Concretamente, los sistemas construidos no tenían capacidad de adaptación, de modo que su funcionamiento y estructura eran la misma para todos sus usuarios. La necesidad de un Sistema de Aprendizaje capaz de ajustar el modo en que los usuarios recorren y leen la información del hipermedia permitió trazar de manera muy precisa el punto de partida de esta investigación.

Objetivo de la tesis

Tal y como se expuso en el proyecto de tesis, nuestro objetivo es abordar la especificación y formalización de un **modelo de adaptación al usuario para el desarrollo de sistemas hipermedia adaptativos, integrales y evolutivos**. Se pretende, además, que el modelo propuesto pueda ser incluido como **Sistema de Aprendizaje del modelo SEM-HP**, estudiando la interacción de éste con el resto de sistemas que lo componen.

La integración del modelo de adaptación en SEM-HP debe ser estudiada desde el punto de vista de la estructura, el funcionamiento y la evolución, de tal forma que los sistemas hipermedia construidos de acuerdo a éste sean capaces de ajustarse a las características y necesidades de cada usuario, y además permitan un proceso de evolución consistente. También se pretende desarrollar un **prototipo**, que permita la creación de sistemas hipermedia conforme al modelo SEM-HP, y que implemente sus características más relevantes.

En concreto, la presente tesis pretende lograr, al menos, un modelo de adaptación con las siguientes características:

- Ser de **aplicabilidad general**, permitiendo desarrollar sistemas hipermedia adaptativos, cualquiera que sea su dominio de conocimiento. De este modo, los sistemas desarrollados podrán utilizarse para propósitos tan distintos como la

creación de manuales educativos o la organización no jerárquica de documentos en un PC.

- Proporcionar el mayor número y variedad posible de **métodos de adaptación** para personalizar tanto el proceso de navegación del usuario como el contenido de los documentos que visita.

- Los sistemas hipermedia desarrollados de acuerdo al modelo deben ser capaces de adaptarse a las características individuales de cada usuario, pero también deben **recoger información sobre el proceso de navegación colectiva** y, basándose en ésta, sugerir modificaciones estructurales que beneficien al grupo de usuarios.

- Fomentar que el autor establezca los **prerrequisitos** de un documento en función de criterios **pedagógicos** como, por ejemplo, las relaciones semánticas de los conceptos que en él se tratan o el conocimiento del usuario sobre otros documentos relacionados, y no como meros mecanismos de ordenación.

- Ser **abierto** y admitir que, en cualquier momento, el autor incluya nuevos documentos en el sistema sin importar cuál sea su origen.

- Ofrecer los mecanismos necesarios para que los sistemas desarrollados sean **adaptables y adaptativos**. De acuerdo a esto, algunas características del modelo de usuario se inicializarán y actualizarán automáticamente, otras necesitarán una intervención explícita del usuario en ambas tareas, y otras serán consultadas inicialmente pero actualizadas automáticamente. Además, se debe conceder cierto grado de **control al usuario** sobre el tipo de adaptación del que desea disfrutar durante su proceso de navegación.

- Tener capacidad para generar dinámicamente **documentos compuestos** a partir de otros documentos.

- Procurar al autor las **herramientas evolutivas** necesarias para realizar, en el momento que lo considere oportuno, modificaciones sobre los elementos del modelo.

- Incorporar un mecanismo basado en **restricciones** que verifique que los cambios introducidos son consistentes y en caso contrario rechace su ejecución.

- Garantizar que el estado del sistema completo es consistente después del cambio, por lo tanto, debe evaluar la repercusión de cada cambio dentro del sistema donde se integra y en los sistemas relacionados, ejecutando la **propagación interna y externa** necesaria.

Estructura de la exposición

La memoria de la presente tesis se divide en seis partes o módulos, que agrupan un total de veinticinco capítulos.

Módulo I – Marco Teórico

Módulo II – El modelo SEM-HP

Módulo III – Sistema de Aprendizaje: Elementos Estructurales

Módulo IV – Sistema de Aprendizaje: Elementos Funcionales

Módulo V – Implementación

Módulo VI – Conclusiones

En el **módulo I** se realiza una revisión y análisis del marco teórico donde se ubica nuestro trabajo. Este módulo se compone de cinco capítulos:

Capítulo 1 – El Conocimiento

Capítulo 2 – Evolución del Software

Capítulo 3 – Sistemas Hipermedia Adaptativos

Capítulo 4 – Sistemas Educativos

Capítulo 5 – Web Semántica

La figura 1 muestra la situación del presente trabajo poniendo de manifiesto la relación que mantiene con los fundamentos teóricos estudiados en este primer módulo. Como se explica más adelante, algunos de éstos han sido incluidos anticipando posibles líneas de trabajo futuro, mientras que otros han sido imprescindibles para el desarrollo del trabajo realizado.

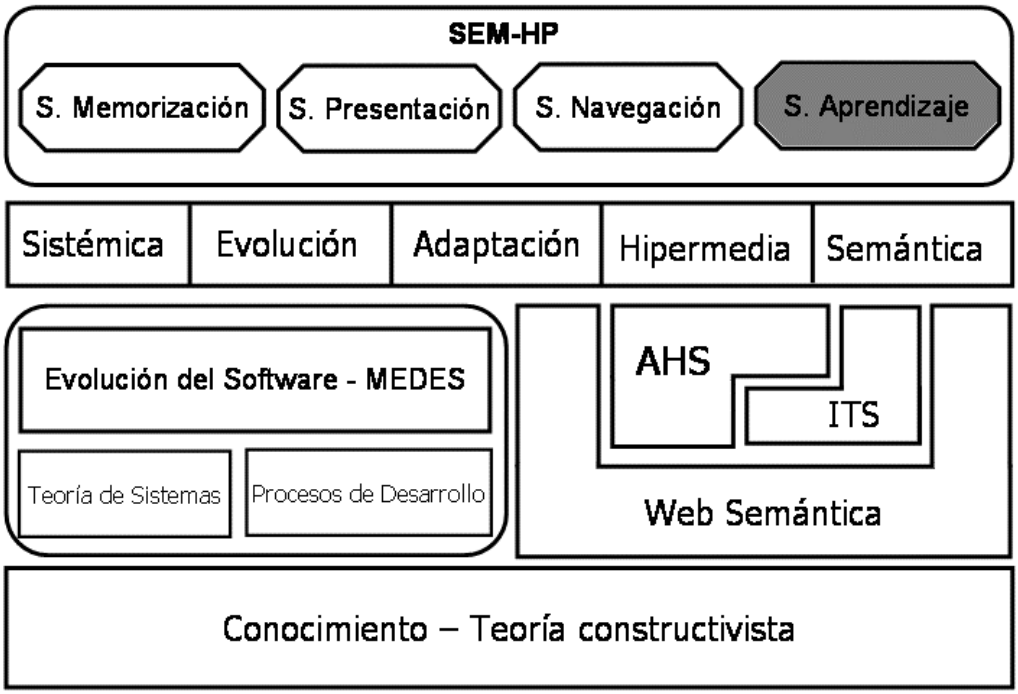


Figura 1: Marco Teórico

En el nivel más bajo se sitúa una visión constructiva del **conocimiento** que constituye un soporte fundamental para poder llevar a cabo las tareas de evolución y adaptación al

usuario. Además, este primer nivel es imprescindible si queremos conseguir que el sistema hipermedia esté basado en un modelo cognitivo donde se haga una representación explícita del conocimiento que aporta [García, 01c].

Respecto a la **evolución**, la visión que se utiliza en este trabajo procede de la metodología MEDES [Parets, 95] según la cuál, un sistema se compone de un conjunto de *sistemas* interrelacionados cuya estructura va madurando a lo largo del tiempo. De manera que, es necesario proporcionar mecanismos evolutivos que prevean esos cambios, y los integren de forma consistente durante el *proceso de desarrollo* del sistema [Parets, 96].

Respecto a la **adaptación del hipermedia**, es preciso estudiar con detalle los trabajos publicados en la literatura científica, especialmente los desarrollados en el área de los sistemas hipermedia adaptativos (**AHS**). A partir de este análisis es posible extraer las características comunes que los identifican, los criterios que los distinguen y sus principales problemas e inconvenientes [Medina, 02].

Debido a la estrecha relación que existe entre los sistemas hipermedia adaptativos y los **sistemas educacionales**, especialmente con los sistemas tutores inteligentes (**ITS**), se examinan también estos últimos, con la intención de anticipar futuras extensiones educacionales del trabajo realizado [Martín, 03][Medina, 04].

Ambos tipos de sistemas encajan en una nueva concepción *web*, la denominada **web semántica** cuyo precursor Tim Berners-Lee define como: "*an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*" [Berners-Lee, 01].

En consecuencia, se estudia este campo, con la intención de dotar al modelo de ciertas características como, por ejemplo, el uso de ontologías, que propicien su integración en el futuro de la web, y la interoperabilidad con otros SHA.

Basándonos en estos fundamentos, se especifica y describe el **Sistema de Aprendizaje** como un modelo de adaptación integral y evolutivo. Dicho sistema, junto con los Sistemas de Memorización, de Presentación y de Navegación, forma parte del **modelo SEM-HP**, de acuerdo al cual es posible desarrollar sistemas *hipermedia adaptativos* y *evolutivos* donde la *semántica* es explícita.

En el **módulo II** se describe el modelo SEM-HP planteando la interrelación del Sistema de Aprendizaje con el resto de sistemas que lo componen, y detallando a nivel conceptual su estructura y funcionalidad. Los capítulos que integran el módulo son dos:

Capítulo 6 – El modelo SEM-HP

Capítulo 7 – El Sistema de Aprendizaje

En la figura 2 se adelanta la arquitectura de doble nivel utilizada en el modelo SEM-HP para llevar a cabo la evolución del software. En el nivel más bajo, el usuario hace uso del sistema hipermedia recorriendo e inspeccionando su dominio de conocimiento. En el nivel más alto, el autor interacciona con el MetaSistema para hacer evolucionar el sistema en el modo deseado.

La figura también exhibe la interrelación que existe entre los cuatro sistemas del modelo. Si nos fijamos en el Sistema de Aprendizaje, nuevamente sombreado en gris, vemos que manifiesta dos tipos de relaciones. Por un lado, su estructura se define sobre determinados elementos del resto de sistemas, lo que motiva que al cambiar éstos deba propagarse el cambio al Sistema de Aprendizaje. Por otro lado, el Sistema de Aprendizaje es capaz de identificar y sugerir modificaciones estructurales, principalmente, a los Sistemas de Memorización y Presentación.

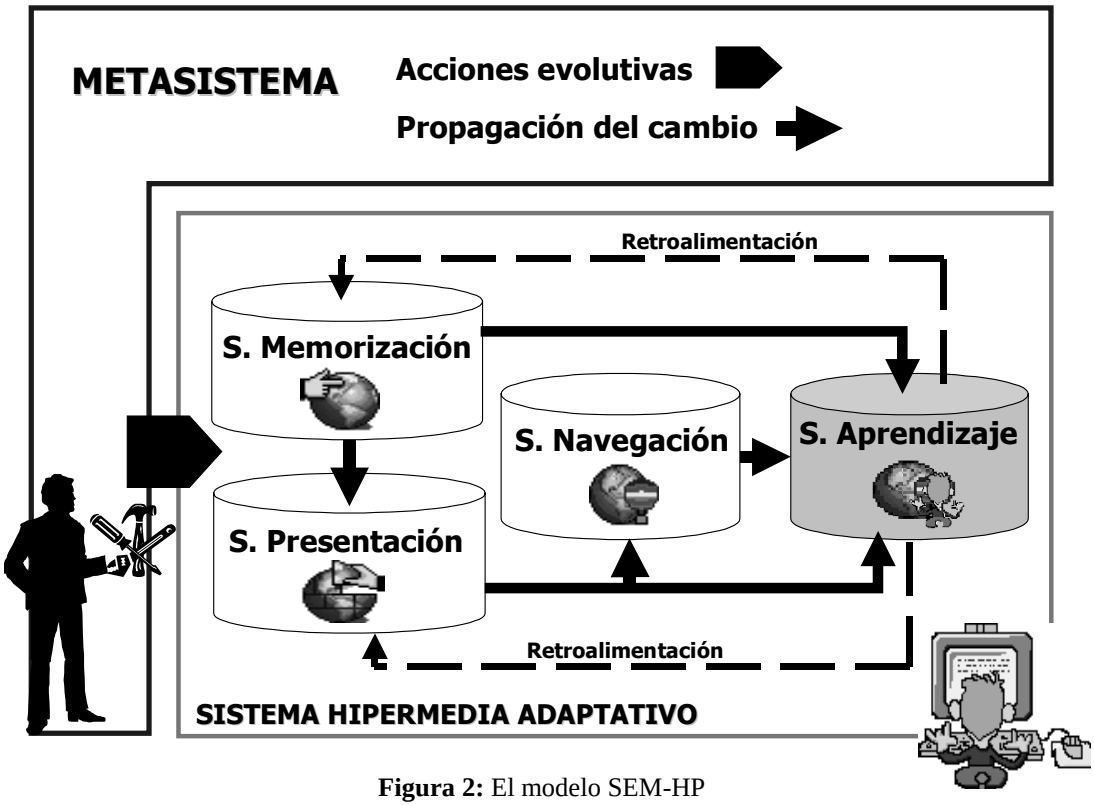


Figura 2: El modelo SEM-HP

En el **módulo III** se especifican, describen y formalizan los elementos estructurales del Sistema de Aprendizaje. Además, se establecen los mecanismos necesarios para comprobar la consistencia estructural del sistema, tanto en sí mismo como en relación a los otros sistemas que integran el modelo SEM-HP. El módulo se divide en seis capítulos:

Capítulo 8 – Reglas de Conocimiento

Capítulo 9 – Reglas de Actualización

Capítulo 10 – Consistencia Rk ∪ Ru

Capítulo 11 – Reglas de Peso

Capítulo 12 – Modelo de Usuario

Capítulo 13 – Propagación Externa del Cambio

Los prerrequisitos pedagógicos que se aconsejan para entender perfectamente la información contenida en cada capítulo son los siguientes:

capítulo 6 → capítulo 8

capítulo 6 ∧ capítulo 8 → capítulo 9

capítulo 8 ∧ capítulo 9 → capítulo 10

capítulo 6 ∧ capítulo 9 → capítulo 11

capítulo 6 ∧ capítulo 8 ∧ capítulo 9 ∧ capítulo 11 → capítulo 12

capítulo 6 ∧ capítulo 8 ∧ capítulo 9 ∧ capítulo 11 ∧ capítulo 12 → capítulo 13

La descripción conceptual del capítulo 7 constituye una ayuda inestimable para comprender más rápidamente los capítulos que componen este módulo.

En el **módulo IV** se detallan y formalizan los elementos funcionales del Sistema de Aprendizaje, destinados a dirigir y adaptar la navegación del usuario en función de la información almacenada sobre éste en su modelo y los elementos estructurales definidos por el autor. Ocho son los capítulos que componen este módulo:

Capítulo 14 – Elección de la EC_A

Capítulo 15 – Modos de Navegación

Capítulo 16 – Navegación por Conceptos

Capítulo 17 – Navegación por Relación Conceptual

Capítulo 18 – Navegación por Conocimiento

Capítulo 19 – Ítems Deseables

Capítulo 20 – Rutas Guiadas

Capítulo 21 – Retroalimentación Adaptativa

Los prerrequisitos pedagógicos necesarios para asimilar de forma correcta la información contenida en cada capítulo son los siguientes:

capítulo 6 ∧ capítulo 12 → capítulo 14

capítulo 8 ∧ capítulo 9 ∧ capítulo 12 ∧ capítulo 14 → capítulo 15

capítulo 9 ∧ capítulo 12 ∧ capítulo 14 ∧ capítulo 15 → capítulo 16

capítulo 9 ∧ capítulo 12 ∧ capítulo 14 ∧ capítulo 15 → capítulo 17

capítulo 8 ∧ capítulo 9 ∧ capítulo 11 ∧ capítulo 12 ∧ capítulo 14 ∧ capítulo 15 → capítulo 18

capítulo 8 ∧ capítulo 9 ∧ capítulo 11 ∧ capítulo 12 ∧ capítulo 18 → capítulo 19

capítulo 8 ∧ capítulo 9 ∧ capítulo 10 ∧ capítulo 11 ∧ capítulo 12 ∧ capítulo 18 → capítulo 20

capítulo 6 ∧ capítulo 14 ∧ capítulo 15 ∧ capítulo 16 → capítulo 21

De nuevo, se recomienda haber leído el capítulo 7 para entender más fácilmente los conceptos formalizados en estos capítulos.

En el **módulo V** hay un único capítulo dedicado a la implementación de un prototipo basado en el modelo SEM-HP.

Capítulo 22 – Implementación

En el **módulo VI** se caracteriza el modelo SEM-HP desde el punto de vista evolutivo y de adaptación al usuario, haciendo especial hincapié en el Sistema de Aprendizaje. También se analizan y comparan algunos trabajos relacionados. Finalmente se enumeran las conclusiones del trabajo realizado y se trazan posibles líneas de trabajo futuro. El módulo tiene tres capítulos:

Capítulo 23 – Consideraciones sobre Evolución y Adaptación

Capítulo 24 – Trabajos Relacionados

Capítulo 25 – Conclusiones y Trabajo Futuro

Al final aparece un **apéndice** que contiene las tablas de acrónimos y abreviaturas, de símbolos, de definiciones y las referencias bibliográficas.

Formato de la exposición

Cada capítulo comienza con una portada que muestra su número y título. En el reverso de ésta se proporciona un breve resumen que indica la información tratada en el capítulo, así como una tabla de contenido. Además, al principio de la memoria se facilita un índice general que desglosa el contenido de cada capítulo. Y antes de cada módulo se presenta una síntesis de contenidos con los capítulos que éste contiene.

Exceptuando títulos, secciones y notas al pie, la memoria de la tesis presenta el formato con el que se escriben estas líneas. No obstante, para facilitar su lectura se utilizan distintos estilos gráficos que identifican visualmente la función del texto.

Def 8.3 [Definición] Las definiciones tienen este aspecto y se numeran consecutivamente dentro de cada capítulo. Por ejemplo, la definición 8.3 sería la tercera definición dentro del capítulo 8. Al final aparece una lista de definiciones ordenada alfabéticamente, donde también se indica el número de la definición.

Las ecuaciones se numeran, entre paréntesis, de uno en adelante en cada capítulo (1)

Los ejemplos se delimitan superior e inferiormente con líneas de borde como las que aquí se utilizan. De esta forma es fácil identificar las partes del texto que contienen una aplicación práctica de los conceptos descritos, una suposición, etc.

Tabla 1: Una tabla

La información que se organiza dentro de una tabla	
	tiene más o menos este formato

Paso 1. Los algoritmos y procedimientos,

Paso 2. Tienen el aspecto,

Paso 3. Que se muestra,

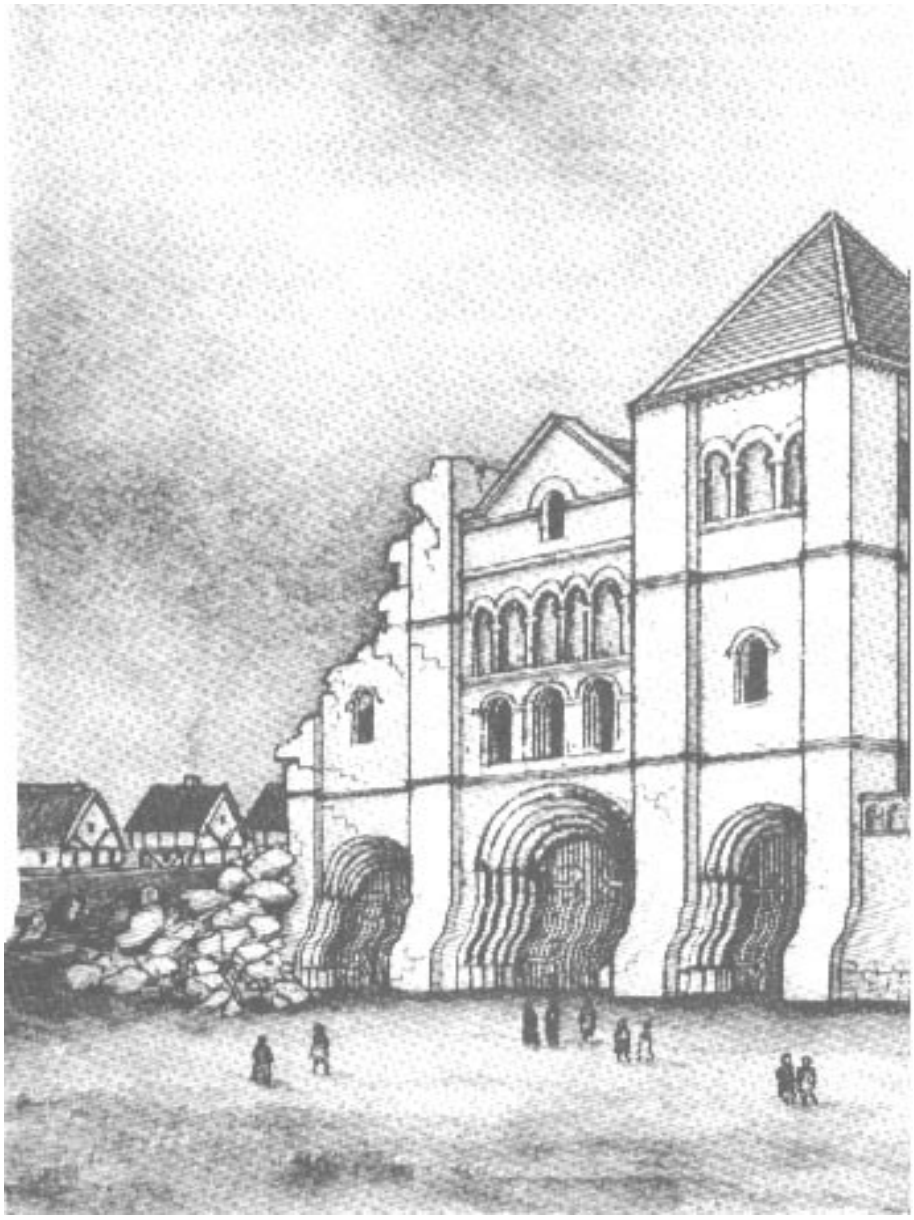
Terminar. En estas líneas.

Algoritmo 1. Un algoritmo

Las referencias se dan entre corchetes y constan del apellido del primer autor y el año en que se publicó el citado artículo. Para referenciar a dos trabajos del mismo autor y año se utilizan letras consecutivas [Medina, 03] [Medina, 03b] [Medina, 03c],...

Módulo I

Marco Teórico

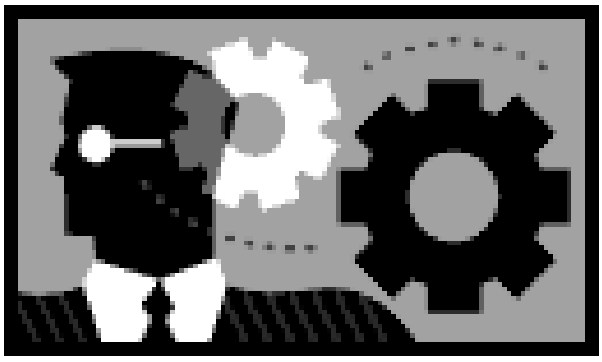


Los Pilares de la Tierra

Módulo I

CAPÍTULO 1

El Conocimiento



Resumen

En este capítulo se plantea la dificultad de establecer una definición precisa para el conocimiento. Se describen algunas características que en la mayoría de los casos acompañan a éste y permiten identificarlo. Y se exponen tres de las teorías más aceptadas acerca de cómo se adquiere el conocimiento. Expresando, finalmente, la visión particular de conocimiento que nosotros adoptamos en la presente tesis.

Tabla de contenidos

El Conocimiento

1. ¿Qué es Conocimiento?

La preocupación por estudiar el conocimiento es casi tan antigua como la filosofía misma. Ya en los diálogos platónicos se pueden encontrar explicaciones del proceso cognoscitivo. Y de forma recurrente distintos filósofos y pensadores se han planteado preguntas como las siguientes: ¿El conocimiento es posible?, ¿es posible conocer el conocimiento?, ¿qué es conocimiento?, ¿cómo podemos estar seguros de nuestro conocimiento?, ¿cómo conocemos?, etc. Dar respuesta a estas y otras cuestiones cruciales entorno al conocimiento son tareas epistemológicas difíciles de resolver de forma consensuada.

Según Dancy [Dancy, 93], epistemología es el estudio del conocimiento y de la justificación de creencia. Por lo tanto es necesario:

1) Encontrar una definición de conocimiento, y

2) Justificar nuestro conocimiento.

De acuerdo a Gettier [Gettier, 63], las anteriores premisas son necesarias pero insuficientes. Según su visión, necesitamos un proceso que nos permita justificar nuestro conocimiento y un modo para generar e incrementar este conocimiento. Pero, además, es necesaria una evidencia de la verdad de nuestro conocimiento, independientemente de nuestro conocimiento y los procesos realizados para su justificación. Esta evidencia puede ser obtenida desde una tabla de verdad.

La noción de verdad es un aspecto importante a la hora de definir conocimiento. Junto a ésta se encuentran otros conceptos como:

- Falsedad (no verdad): Nada puede conocerse si es inferido desde una creencia falsa o un grupo de creencias donde al menos una es falsa.

- Refutabilidad: La justificación debe ser irrefutable, es decir la adicción de una nueva verdad no puede impugnarla. En otro caso, nunca podríamos conocer nada porque siempre podemos esperar una nueva verdad.

- Fiabilidad: Una creencia justificada puede ser conocimiento si ésta se deriva usando un método fiable.

Según la doctora Ballesteros-Olmos [Ballesteros, 03], el conocimiento es un fenómeno que se produce siempre dentro de una relación entre dos miembros o elementos: el sujeto que conoce y el objeto conocido. Ambos son necesarios. Si falta uno no hay conocimiento. En general se piensa que la conciencia del sujeto se comporta de forma activa en el conocimiento; es la que capta al objeto. En la conciencia o mente del sujeto se configura la representación del objeto; si esa representación coincide con la realidad hay conocimiento. “El conocimiento y la verdad dependen de la adecuación entre el intelecto y la cosa de la realidad”.

2. CARACTERÍSTICAS DEL CONOCIMIENTO

Desde una perspectiva menos ambiciosa, que no pretende definir conocimiento ni averiguar los métodos empleados para adquirirlo, podemos enunciar una serie de características del conocimiento.

El conocimiento es una capacidad humana y no una propiedad de un objeto como pueda ser un libro. Es esencial distinguir la diferencia entre datos, información y conocimiento, tres conceptos a menudo utilizados erróneamente como sinónimos. Los **datos** son resultados que podemos medir. La **información** es obtenida después de enlazar y estructurar convenientemente los datos en función de su significado. Para que dicha información se convierta en **conocimiento** aprovechable para la persona, es necesario que realice un proceso de interpretación y análisis. Una vez adquirido, el conocimiento capacita a la persona para responder anticipadamente y de forma correcta a una situación futura. Una definición táctica, establece el conocimiento como la capacidad para convertir datos e información en acciones efectivas [DAEDALUS, 04].

El conocimiento genera conocimiento mediante la utilización de la capacidad de razonamiento o inferencia. De este modo, el conocimiento no permanece estático, sino que se mueve, es decir, es transmitido, transformado o ampliado. La transmisión del conocimiento implica un proceso intelectual de enseñanza y aprendizaje. Obviamente, transmitir una información es fácil, mucho más que transmitir conocimiento. El conocimiento es a menudo dependiente del contexto del que aprende, por lo tanto, el trasmisor debe conocer el contexto o el modelo del mundo del receptor.

El conocimiento tiene estructura y es elaborado, implica la existencia de redes semánticas ricas en entidades abstractas o concretas y con múltiples relaciones entre ellas. Una base de datos, por muchos registros que contenga, no constituye conocimiento. El conocimiento representado en estas redes será fundamentalmente explícito, aunque, también es posible incluir conocimiento heurístico resultado de la experiencia del individuo. La representación del conocimiento puede ser formal o informal siempre y cuando sea inteligible para el receptor.

3. ¿CÓMO CONOCEMOS?

La respuesta a esta pregunta plantea actitudes muy diferentes. Entre las más destacadas encontramos tres enfoques o teorías:

- Teoría Fundacionalista: Define y justifica el conocimiento individual. Se caracteriza por su positivismo y realismo.
- Teoría de la Coherencia: Relacionada con el enfoque sistémico mantiene un modo holístico de concebir la justificación y la adquisición de conocimiento.
- Teoría del Constructivismo: Más preocupado por el proceso de constitución del conocimiento que en sus fundamentos.

3.1 Teoría Fundacionalista

Descartes [Descartes, 1596-1650] es considerado el padre del racionalismo fundacionalista en la medida que inicia la duda metódica renegando de su propio

aprendizaje en dialéctica, historia y letras producto de su formación escolástica, y su crítica radical al seudoconocimiento que aportan los sentidos.

El fundacionalismo postula que estamos en presencia de un conocimiento fundamental cuando existe una creencia básica incorregible.

> "... advertí en seguida que aún queriendo pensar de este modo que todo es falso, era necesario que yo, que lo pensaba, fuese alguna cosa. Y al advertir que esta verdad –pienso, luego existo– era tan firme y segura que las suposiciones más extravagantes de los escépticos no eran capaces de conmoverla, juzgué que podía aceptarla sin escrúpulo como el primer principio de la filosofía que buscaba." Descartes citado en [Echeverría, 88].

Esta teoría supone la existencia de fundamentos para el conocimiento, una base indubitable de piezas de conocimiento, que puedan reconocerse infaliblemente como tales. El conocimiento es un edificio en el que las creencias derivadas descansan sobre otras indudables y directas, y se transforma, finalmente, en hábitos instintivos e innatos.

Según Descartes, debemos empezar intentando dudar de todas nuestras creencias con la esperanza de descubrir así su núcleo indubitable. Una vez obtenidas las fuentes infalibles de conocimiento, una proposición puede ser verificada de modo concluyente. Las cuestiones del "punto de partida" y del modelo de inferencia constituyen el problema central de esta teoría. La concepción fundacionalista de la mente fija verdades universales y considera la realidad en términos de verdadero-falso.

Este enfoque concibe dos niveles (véase la figura 1). El primero está constituido por los fundamentos epistemológicos o creencias básicas que proceden de nuestro estado sensorial o nuestra experiencia más cercana. Producen sentencias observables, que no se pueden inferir y que resultan infalibles. Sobre este nivel se sustenta la superestructura compuesta de creencias no básicas inferidas a partir de las creencias básicas del nivel inferior. Producen sentencias no observables.

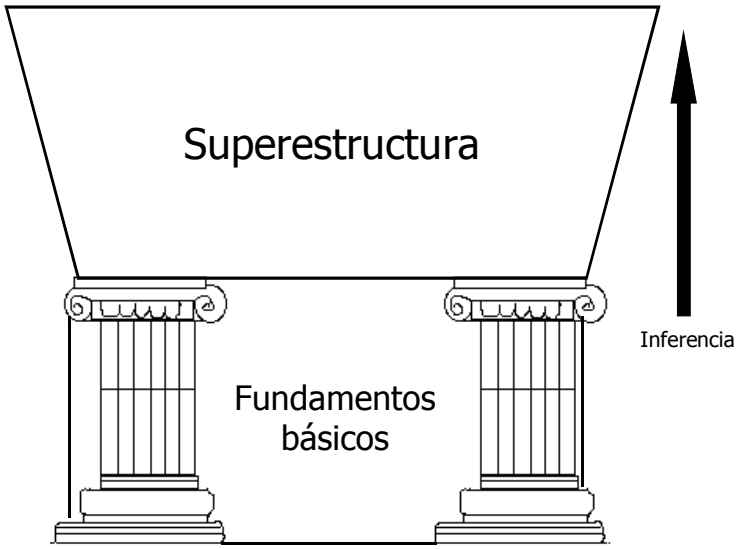


Figura 1: Efnfoque Fundacionalista

Esta teoría acepta tres asunciones:

1) Las creencias básicas son ciertas.

2) Los principios básicos son hechos y datos incorregibles e indudables.

3) Empirismo radical.

El proceso de justificación es unidireccional: la evidencia confirma o invalida teorías, pero la teoría no puede confirmar o invalidar evidencias. Sin embargo, la historia de la ciencia muestra que no hay sentencia inmune a revisión.

3.2 Teoría de la Coherencia

Según este enfoque, no existe un conjunto de fundamentos inamovibles sobre los que se construye el conocimiento. Lo que vuelve verdaderas las creencias son otras creencias (una evidencia), no la estimulación sensorial, el mundo.

Esta teoría se utiliza desde el siglo dieciocho, y la han sostenido filósofos racionalistas y empiristas lógicos. Se ha utilizado sobre todo en ciencias formales, como matemáticas y lógica. Esta teoría supone que la realidad es un todo coherente y que los hombres a partir de su estructura intelectual la pueden captar de esa manera. Los requisitos que deben cumplir las proposiciones para ser verdaderas en esta teoría son dos:

1) No pueden oponerse entre sí, al contrario; una proposición debe apoyar o apuntalar a la otra, esto se llama requisito de la no contradicción, y

2) Cada proposición debe derivarse de alguna otra o servir como base para que se deriven otras a partir de ellas. Estos dos requisitos se cumplen en los sistemas axiomáticos de la matemática y de la lógica actual.

Según esta teoría, no hay una verdad parcial, sino la verdad es un todo, y un juicio particular puede pretender validez sólo si es coherente con el resto de los juicios que integran la totalidad de la verdad. Esto es, la coherencia se determina por el posicionamiento de la parte en el todo, en el sistema.

La coherencia es una propiedad de un conjunto de creencias. Requiere consistencia de las creencias implicadas e incrementalidad, es decir, el conjunto de creencias puede crecer. Una clara diferencia con el anterior enfoque, es la explicación mutua entre las creencias. Así, cuando el conjunto de creencias crece cabe esperar que cada miembro del conjunto esté mejor explicado por el resto. Es una explicación bidireccional.

La noción de verdad define una proposición como cierta si es miembro de un conjunto coherente. La noción de justificación establece que si el conjunto de creencias de *a* es más coherente con la creencia *p* que sin ella o con otra creencia, *a* será justificación para creer *p*.

El empirismo en esta teoría es más moderado que en el enfoque fundacionalista. El conocimiento no sólo se incrementa por mecanismos de inferencia y justificación. Y se admite el conocimiento colectivo como una fuente de conocimiento. Introduciendo la

idea de conocimiento como un fenómeno social, algo que puede ser compartido e incrementado como consecuencia.

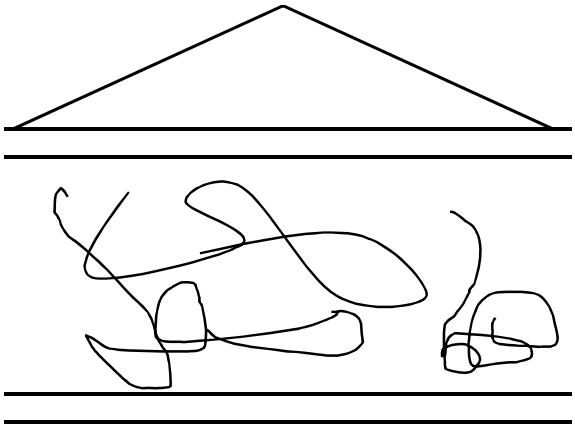


Figura 2: Metáfora de coherencia

3.3 Teoría del Constructivismo

El planteamiento base en este enfoque es que el individuo es una construcción propia que se va produciendo como resultado de la interacción de sus actitudes internas y el medio ambiente en que se encuentra inmerso. Así, el conocimiento no es una copia fiel de la realidad, sino una construcción que hace la persona misma [Carretero, 94]. Este enfoque está muy cerca de la teoría de la coherencia y asigna un papel activo al sujeto que conoce.

El aprendizaje no es una tarea simple de transmisión y acumulación de conocimientos sino un proceso activo, en el que el individuo debe integrar, extender, restaurar e interpretar la información que recibe para construir su conocimiento. Se establece, por tanto, un valor social e individual del conocimiento de acuerdo a un pragmatismo socio-cultural. El individuo es parte esencial del proceso de adquisición del conocimiento, tal y como se muestra en la metáfora del constructivismo de la figura 3.

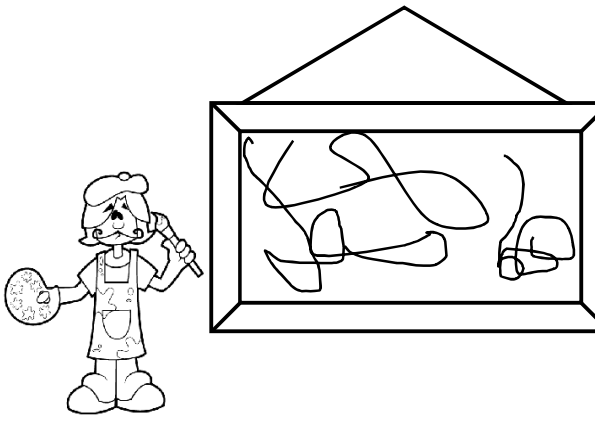


Figura 3: Metáfora del constructivismo

El sujeto construye estructuras a través de la interacción con su entorno y los procesos de aprendizaje. El conocimiento y su representación se hace inseparable: conocer es

representar una realidad usando algún sistema de símbolos. Por este motivo, los diseñadores de materiales didácticos deben prestar especial interés a la forma de organizar la información y los profesores fomentar el desarrollo de estructuras que faciliten futuros aprendizajes del alumno.

La teoría del constructivismo [Le Moigne, 95] rechaza la necesidad de aprender en un orden estricto. Las estructuras cognitivas son las representaciones organizadas de experiencias previas. Mientras captamos información estamos constantemente organizándola en unidades que llamamos estructuras y que generalmente se asocian con información ya existente en otras estructuras.

En este enfoque se enfatiza la construcción de nuevo conocimiento y maneras de pensar mediante la exploración y la manipulación activa de objetos e ideas, tanto abstractas como concretas. Los mapas conceptuales, utilizados como herramienta para navegar y organizar información, proveen un entorno sobre el cual es posible construir ambientes educativos basados en un enfoque constructivista [Cañas, 00].

El enfoque constructivista distingue tres elementos inseparables: conocimiento (el resultado), cognición (el proceso) e inteligencia (el autor). La **cognició**n es irreversible, dialéctica y recursiva. Y el **conocimiento** es más que un proceso de adquisición: cambia la inteligencia, cambia el mundo, y por lo tanto, tiene una responsabilidad social. Desde una perspectiva teológica se plantea la pregunta, ¿cuál es la finalidad del conocimiento?.

4. NUESTRA VISIÓN DEL CONOCIMIENTO

Nosotros adoptamos una visión epistemológica coherentista y constructiva, por lo tanto relativista y restrictiva si se me permite. No suponemos un conjunto de verdades iniciales, sino que concebimos la existencia de múltiples relaciones entre la información proporcionada. De modo que al aprender sobre un trozo de información se aumenta, también, el conocimiento sobre otras piezas de información relacionadas. De esta manera, el conocimiento se construye poco a poco en base a las interrelaciones existentes entre la información consultada. Obviamente el sujeto que aprende es un factor esencial en el proceso de aprendizaje, ya que las características de este proceso dependerán de las actitudes, experiencia previa e intereses concretos de cada individuo.

Además no creemos que exista un orden estricto para conocer, pero sí que para adquirir un determinado conocimiento es necesario estar en disposición de un conocimiento previo. Para obtener este conocimiento se pueden proporcionar multitud de combinaciones de información. Esto es, no existe una única secuencia de trozos de información cuyo estudio proporcione el conocimiento requerido, sino varias.

De esta forma, para que la lectura de un trozo de información se pueda transformar en conocimiento será necesario que el lector haya adquirido previamente un conocimiento más o menos profundo de otra información relacionada, no importa cuál sea la combinación seguida para alcanzarlo. En el caso de trozos de información muy básicos se puede presuponer que el individuo posee el conocimiento elemental requerido para comprender dicha información.

Puesto que el conocimiento es algo que el individuo construye poco a poco, hablaremos de **estado de conocimiento** como la configuración de conocimientos que el sujeto posee en un determinado instante. El cambio de estado se produce cada vez que se añade o mejora algún conocimiento de la configuración. Este cambio de estado es activado directamente por el usuario como consecuencia de estudiar un trozo de información, siempre y cuando esté en condiciones de extraer conocimiento de la lectura. Esto es, sólo si su estado de conocimiento antes de acceder a esa información engloba los conocimientos necesarios para asimilarla.

CAPÍTULO 2

Evolución del Software

Resumen

La evolución del software es una tarea fundamental si de desea hacer un uso correcto de éste durante un largo tiempo. En el presente capítulo se describen algunas de las situaciones más frecuentes del cambio, se definen los conceptos de “teoría de sistemas” y “modelo”, y en base a éstos se enmarca el termino “evolución”. Además, se resaltan las principales características de la evolución y se presentan dos taxonomías para clasificar sus distintos tipos. Defendiendo la evolución frente al mantenimiento y apostando por una concepción evolutiva del software como un proceso de maduración, materializado en el modelo MEDES.

Tabla de contenidos

Evolución del Software

1. INTRODUCCIÓN

Debido a que el software sufre continuos cambios durante su diseño y construcción, así como durante su uso, la evolución del sistema es un problema crucial en el proceso de desarrollo software.

Existen motivos muy diversos por los que se hace necesario realizar cambios en un producto software. Por ejemplo, frecuentemente se encuentra como origen del cambio, un desacertado proceso de comunicación entre el usuario y el equipo de desarrollo, que obligará en un futuro a replantear los requisitos del sistema. Además, no es extraño que las necesidades y prioridades de los usuarios cambien a medida que se desarrolla el producto, y a menudo, surgen variaciones no esperadas en el entorno del sistema. Por otro lado, el conocimiento del sistema se incrementa conforme se desarrolla el producto, con lo que se irá detallando la especificación y concretando su estructura.

Así pues, cada día es más indiscutible la necesidad de establecer unas sólidas bases metodológicas, que permitan la creación de herramientas que ayuden a desarrollar software con capacidades evolutivas. A su vez, esto implica que sea conveniente aportar modelos de desarrollo capaces de integrar el continuo cambio sufrido por este tipo de sistemas.

Para poder enmarcar convenientemente el término evolución aplicado a sistemas software y posteriormente reflexionar sobre las distintas metodologías y herramientas que hoy en día permiten el desarrollo evolutivo del software, es necesario introducir previamente dos conceptos básicos, el de **sistema** y el de **modelo**.

2. TEORÍA DE SISTEMAS

La teoría de sistemas es uno de los pilares en los que se fundamentan los modelos evolutivos. No obstante, han sido muchos los autores y de muy diversas áreas los que han mostrado su preocupación por este tema. De su trabajo, podemos extraer las siguientes definiciones para el término **sistema**:

Tabla 1: Noción de Sistema

Autor	Definición de Sistema
Bertalanffy	“Complejo de elementos en interacción” [Bertalanffy, 68]
	“Conjunto de elementos que se relacionan entre ellos y con el medio" [Bertalanffy, 75]
	En un sistema "las propiedades y modos de acción de los niveles superiores no pueden explicarse por la suma de las propiedades y modos de acción que corresponden a sus componentes considerados aisladamente” [Bertalanffy, 75]
	“Correlato conceptual de ciertos rasgos universales de objetos observados...” [Bertalanffy, 75]
Le Moige	“Algo, que dentro de algo, para algo, hace algo, por algo, que se transforma en el tiempo” [Le Moigne, 77]

	"La representación de un fenómeno activo, percibido, identificable por sus proyectos en un entorno activo, en el que funciona y se transforma teleológicamente" [Le Moigne, 90]
Morin	"Unidad global organizada de interrelaciones entre elementos, acciones o individuos" [Morin, 77]
	"Concepto complejo de base que concierne a la organización" [Morin, 77]
Lindgreen	"Concepción de un área llamada dominio del sistema en la que los elementos se consideran relacionados formando un todo, que se concibe como poseedora de, al menos, una propiedad sistémica particular en relación con el entorno, propiedad que no es poseída por ninguno de los elementos" [Lindgreen, 90]

Aunque son muy numerosas las definiciones que encontramos en la literatura científica referidas al término sistema, la mayoría de éstas aceptan ampliamente las consideraciones de Le Moigne acerca de lo que no es un sistema. En ellas establece que un sistema no puede ser analizado simplemente por enumeración de sus elementos y de sus relaciones [Le Moigne, 77]. Y que los sistemas reales son sistemas complejos, y por lo tanto, irreducibles a un modelo único y completamente calculable [Le Moigne, 90].

3. NOCIÓN DE MODELO

El término modelo resulta lo suficientemente ambiguo como para permitir su utilización en contextos muy dispares, como el modelo de sociedad, el modelo atómico o el modelo de ciclo de vida del software. Por lo tanto, es necesario concretar el significado de este término dentro de la teoría de sistemas y la evolución del software.

- Siguiendo el enfoque general de Bertalanffy [Bertalanffy, 75] se puede considerar toda teoría científica como "un modelo conceptual, donde se reflejan clara y esquemáticamente ciertos aspectos de un fenómeno natural y es posible hacer deducciones y predicciones comprobables".
- De acuerdo a la concepción más estricta presentada por Martínez y Requena [Martínez, 86], podemos considerar un modelo como la representación formal de un sistema, mediante un lenguaje de cualquier índole, desde el lenguaje natural al matemático.

Respecto a las propiedades fundamentales que debe manifestar todo modelo, describiremos de forma escueta las siguientes:

- El modelo no suele ser idéntico ni completo respecto a la entidad modelada, puesto que su objetivo es realzar ciertas características de ésta, deben existir diferencias con el estado original de las cosas.
- El modelo es una creación conceptual libre, que no puede derivarse simplemente de los hechos experimentados u observados. Esto es, el modelado no es objetivo, sino proyectivo.
- Un mismo sistema puede representarse mediante modelos muy distintos, elaborados con criterios y finalidades diversas.
- No existe ningún método para la validación universal de un modelo, ya que no sólo existe pluralidad de modelos concebibles de un mismo fenómeno, sino también de métodos para su modelado.

- El grado de abstracción de un modelo evoluciona en el tiempo.
- El nivel descriptivo de un modelo es siempre relativo, ya que cualquiera que sea el modelo teórico que se considere, siempre se puede intentar evidenciar niveles explicativos más profundos de los que derivarlo y viceversa.

Por último, la tabla 2 incluye una clasificación que agrupa los modelos según la finalidad con la que son construidos.

Tabla 2: Caracterización de modelos según su finalidad

Modelo	Finalidad
Explicativo	Aclarar las relaciones entre un conjunto de fenómenos estableciendo analogías con un fenómeno ya conocido.
Predictivo	Realizar predicciones acerca del comportamiento de los fenómenos.
Ejemplo	Presentar los aspectos relevantes que deben considerarse para repetir, con éxito, un determinado conjunto de fenómenos.
Simulativo	Reproducir artificialmente un fenómeno para observar su comportamiento bajo condiciones variables y controladas.
Constructivo	Permitir la producción de un modelo material.
de Decisión	Extraer la información relevante para realizar una modificación del fenómeno modelado.

Centrándonos en el concepto de sistema software que es el que principalmente nos concierne, usaremos la definición de Parets [Parets, 95], la cual concibe un sistema software como un "modelo simbólico, ejecutable en un ordenador, de parte del comportamiento de un sistema de información, que posee una organización intrínseca activa, constituida por una red de procesadores en interacción entre sí y con el entorno".

4. MODELADO DEL PROCESO EVOLUTIVO

Desde el punto de vista del modelador, la evolución de un sistema software se puede concebir como un cambio en las propiedades de éste de acuerdo con los requerimientos de su entorno. De este modo, para modelar la evolución, será necesario representar el proceso de cambio en las propiedades del propio sistema.

La dificultad radica en que no es posible realizar una formulación genérica de las propiedades que debe estudiar el modelador, ni siquiera de qué cambios en estas propiedades son, a priori, considerables como evolutivos. Sin embargo, podemos aceptar algunas formulaciones genéricas de gran utilidad en la representación de la evolución. Con este fin, esbozamos a continuación ciertas características que para nosotros, marcan el sentido específico de la concepción evolutiva y que permitirán matizar, posteriormente, el proceso evolutivo del desarrollo de software.

Para que sea posible realizar un proceso de evolución efectivo, es necesario que el sistema sea un **sistema abierto**, esto es, que esté en continuo intercambio e interacción con su entorno, de manera que pueda identificar las transformaciones sufridas por éste. Esto permite que el sistema no se equilibre con referencia a un entorno, sino que realmente evolucione con él.

La evolución es un **proceso irreversible, impredecible y no determinista**. Su curso es lógico y comprensible, pero no está predeterminado, y por lo tanto, no es previsible. Además, no es un proceso continuo, en muchos terrenos de la ciencia empírica hay numerosas pruebas de que los sistemas dinámicos no evolucionan uniforme y continuamente al paso del tiempo, sino a saltos relativamente bruscos y repentinos [Laszlo, 87].

En tanto la evolución supone un cambio es necesario que el sistema de representación defina su identidad. De modo que exista, alguna **propiedad** del sistema que se conserve **invariante** a través de sus transformaciones. Aunque estos invariantes serán relativos, en la medida en que ninguna característica de un sistema se conserva un tiempo infinito en un entorno dado [Walliser, 77].

La evolución es un **proceso** intrínsicamente **complejo**, tal y como se deduce de la historia de la evolución biológica [Torres, 99] o social, que por supuesto requiere **autonomía** y **autoorganización**. Es cierto que evolución y finalidad están íntimamente ligadas, sin embargo, en determinados sistemas, es posible que la evolución suponga un replanteamiento de las finalidades para adecuarse al medio. Esto depende del tipo del sistema y la rigidez de sus comportamientos.

5. TAXONOMÍAS DE LA EVOLUCIÓN DEL SOFTWARE

En esta sección se exponen dos recientes taxonomías, la primera de Massimo Felici, que propone un marco conceptual para analizar los diferentes fenómenos evolutivos que tienen lugar en un sistema software. Y la segunda propuesta por Mens, Buckley, Zenger y Rashid, se basa en los mecanismos de cambio y en los factores que influyen en dichos mecanismos.

5.1 Taxonomía de la evolución de Massimo Felici

La taxonomía de Massimo Felici [Felici, 03], establece dos dimensiones en el espacio evolutivo: **la dimensión temporal** y **la física**. La evolución puede darse a lo largo de todas las fases por las que pasa un sistema software durante su vida útil. Por ello, la dimensión temporal recoge desde la evolución en el diseño a la evolución en el uso. Por su parte, la dimensión física abarca desde la evolución física (*hard evolution*) hasta la evolución lógica (*soft evolution*).

Como puede verse en la figura 1, dentro de este espacio se sitúan los principales fenómenos evolutivos:

- **Evolución del software**, teniendo en cuenta la evolución desde el punto de vista del producto.
- **Evolución en la arquitectura**, desde el punto de vista del nivel de diseño. Distingue evolución de las arquitecturas, de los componentes y de ambos.
- **Evolución de los requisitos**. Los requisitos se usan como medio de interacción y por ello representan un lugar natural donde capturar información sobre la evolución de los sistemas software.
- **Evolución de los sistemas software**, teniendo en cuenta el punto de vista sistémico que enfatiza los factores humanos con respecto a la evolución.

- **Evolución organizativa**, que enfatiza la interacción entre los sistemas software y su entorno.

Estos fenómenos no son excluyentes, de modo que pueden superponer unos sobre otros.

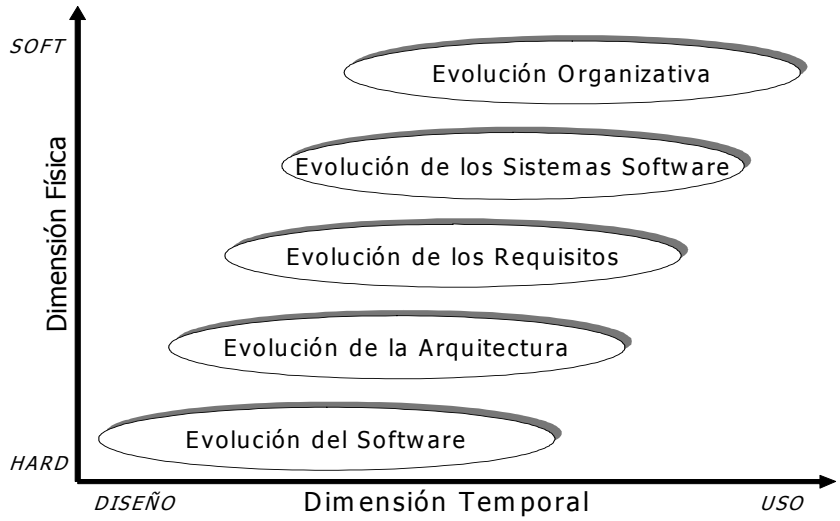


Figura 1: Taxonomía de Massimo Felici [Felici, 03]

5.2 Taxonomía de la evolución de Mens, Buckley, Zenger y Rashid

La idea de esta taxonomía [Mens, 03] es proporcionar un marco conceptual para encuadrar herramientas concretas, formalismos y métodos dentro del dominio de la evolución software. Debido a su importante aumento se hace necesario un vocabulario común que permita categorizar y comparar la evolución que soportan las distintas herramientas y técnicas.

Como puede observarse en la figura 2, la taxonomía propone cuatro grupos lógicos para analizar **cuándo, dónde, qué y cómo tienen lugar los cambios** evolutivos. Dentro de cada grupo se recogen una serie de propiedades, que son descritas muy brevemente en adelante.

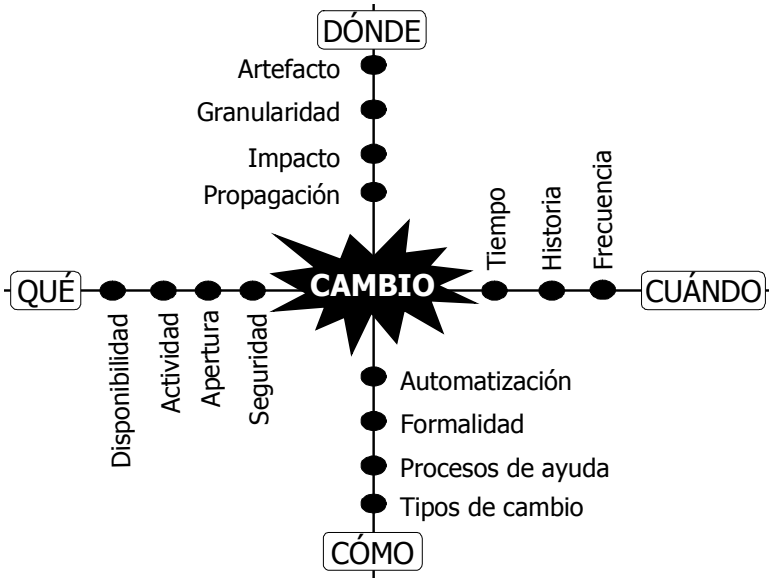


Figura 2: Taxonomía de Mens, Buckley, Zenger y Rashid

Propiedades temporales (**cuándo**):

- Tiempo del cambio: Dependiendo del lenguaje de programación, es posible distinguir en qué fase del ciclo de vida se produce el cambio.
  - Tiempo de compilación: Evolución estática.
  - Tiempo de ejecución: Evolución dinámica.
  - Tiempo de carga.
- Historia del cambio: Se refiere a todos los cambios, secuenciales o paralelos, que han tenido lugar.
  - Sin historia explícita.
  - Herramientas de control de versiones.
    - Tipos de versionado:
      - Estático.
      - Dinámico: Pueden coexistir varias versiones en tiempo de ejecución.
      - Completo: Permite que componentes de diferentes versiones se ejecuten simultáneamente.
    - Realización de cambios:
      - Secuencial.
      - En paralelo: Varias personas pueden realizar cambios sobre los mismo datos al mismo tiempo. Puede ser síncrono o asíncrono.
- Frecuencia del cambio: Continuamente, periódicamente o arbitrariamente.

Objetos que cambian (**dónde**):

- Artefacto: Elementos y partes del sistema que se modifican.
- Granularidad: Escala de los artefactos que cambian. Puede ser fina, media o gruesa.
- Impacto del cambio: Puede ser desde un impacto local hasta un impacto extendido a todo el sistema.
- Propagación del cambio.

Propiedades del sistema (**qué**):

- Disponibilidad: Los sistemas que deben estar siempre disponibles deben ser capaces de evolucionar en tiempo de ejecución.
- Actividad:
  - Reactivos: Los cambios son originados externamente.
  - Proactivos: El sistema produce cambios automáticamente sobre sí mismo.
- Apertura: Se distinguen los sistemas abiertos de los cerrados, según provean o no marcos para facilitar sus posibles extensiones.
- Seguridad:
  - Evitar un comportamiento erróneo tras la evolución.

- Seguridad estática (en tiempo de compilación).
- Seguridad dinámica (en tiempo de ejecución).

➢ Nivel de seguridad: Protección contra virus, prevención de accesos no autorizados, detectar inconsistencias de representación, etc.

Soporte del cambio (**cómo**):

- Grado de automatización: Cambio total o parcialmente automatizado.
- Grado de formalidad: Cambios implementados *ad-hoc* o basándose en algún formalismo matemático.
- Procesos de ayuda que faciliten las actividades de cambio.
- Tipo de cambio:
  - ➢ Estructural.
  - ➢ Funcional.

6. CARENCIAS DE LOS MÉTODOS DE DESARROLLO SOFTWARE

Desde la Ingeniería del Software [Fairley, 87] se propone la aplicación sistemática de métodos para la realización del sistema software, dentro de un determinado marco temporal definido por un modelo de ciclo de vida.

Durante este tiempo, han sido varios los modelos de ciclo de vida que se han propuesto para definir las distintas fases a través de las que se mueve un proyecto de desarrollo software. El primer ciclo de vida surge a finales de los 70 y es propuesto por Winston Royce. Se trata de un modelo que determina etapas muy rígidas y bien definidas y es conocido con el nombre de modelo en cascada [Royce, 70] (véase la figura 3).

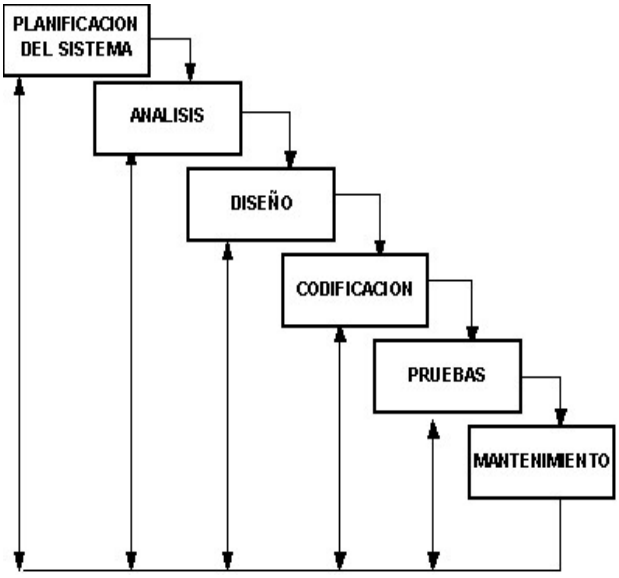


Figura 3: Modelo en cascada

Sin embargo, es un hecho que los requerimientos y funcionalidades de un sistema de software evolucionan, cambian y se modifican constantemente a medida que el producto se está construyendo, por ello se hace necesario el uso de esquemas de desarrollo más

flexibles de tipo espiral [Boehm, 88]. Estos modelos de ciclo de vida (véase la figura 4) son mucho más dinámicos y promueven un desarrollo incremental a través de prototipos.

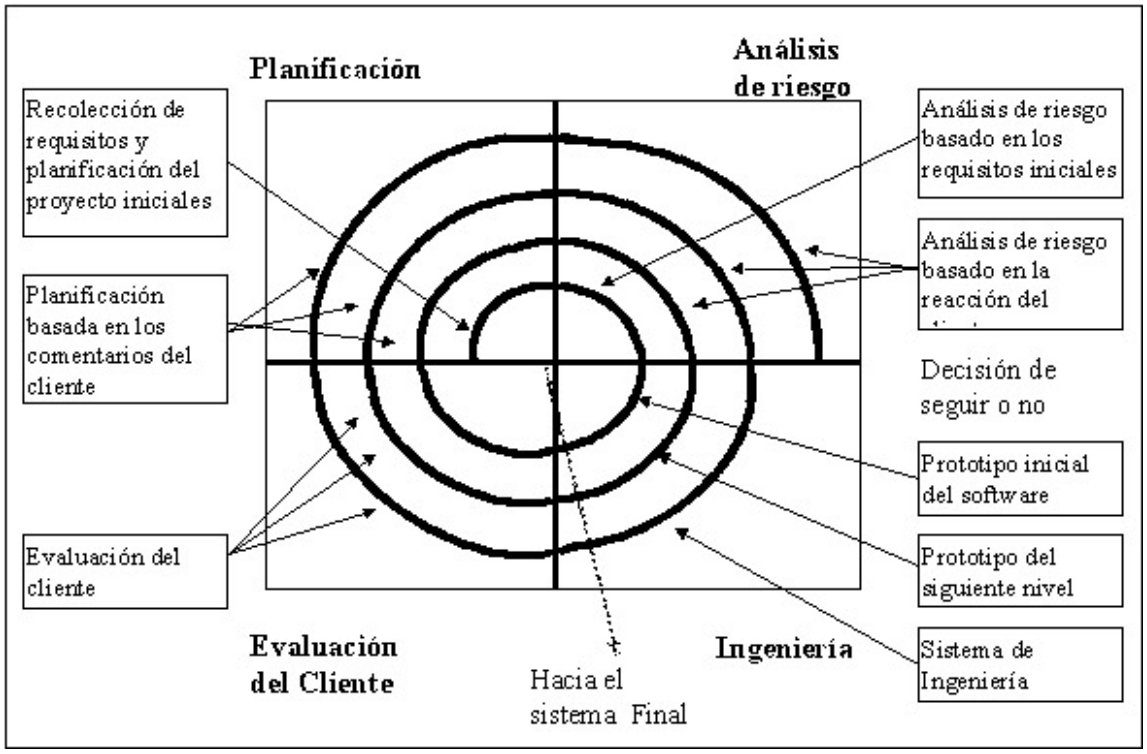


Figura 4: Modelo en espiral

En [Parets, 93] se realiza un estudio comparativo de los métodos de especificación de requerimientos y diseño más populares en Ingeniería del Software dentro del marco de la Teoría del Sistema General. A partir de éste, se deduce que tanto los sistemas software como los métodos de desarrollo adolecen de numerosos problemas cuando pretenden ser utilizados como herramientas de concepción. Entre estos problemas podemos destacar los siguientes:

- El proceso de conocimiento es un proceso creativo, y por lo tanto, difícil de ajustar a un esquema perfectamente establecido. Como tal requiere numerosas iteraciones de desatino, refinamiento y creación hasta llegar a una estructuración aceptable. A pesar de ello, los métodos existentes no recogen la iteratividad, recursividad y complejidad de este proceso.
- Los métodos estudiados consisten en una serie de etapas que deben seguirse fielmente y unos niveles de abstracción que se derivan jerárquicamente cada uno del anterior. Sin embargo, el núcleo del método debería estar constituido por conceptos y herramientas de representación, de modo que el propio método proporcione los instrumentos para la evolución del modelo.
- Por último y no menos importante, ni la dinámica del sistema ni la actividad del modelador forman parte de los métodos. Las herramientas de las que disponen no permiten reflejar los cambios acaecidos en el sistema, y por lo tanto, no contemplan la capacidad evolutiva del sistema ni el papel del modelador.

7. EVOLUCIÓN FRENTE A MANTENIMIENTO

Para algunos autores [Rodríguez, 01], históricamente, el mantenimiento puede ser considerado la primera forma de evolución software. Tradicionalmente, el mantenimiento se ha considerado la última etapa del ciclo de vida del software y debido a su elevado costo se ha convertido en el centro de interés de múltiples investigaciones. Sin embargo, el mantenimiento comienza cuando ya se ha iniciado el funcionamiento del sistema software y es una etapa aislada donde, probablemente, es necesario parar el sistema para poder realizar sobre él las operaciones de mantenimiento necesarias.

Sin embargo, es más realista considerar la Ingeniería del Software como un proceso evolutivo, en el cual el software es modificado continuamente durante su periodo de vida como respuesta a los requisitos cambiantes y a las necesidades del usuario. Desde esta perspectiva [Parets, 96], desarrollo y mantenimiento no son dos procesos separados. Esto es, la evolución concibe la construcción y funcionamiento del software como un proceso continuo.

Existen tendencias muy diferentes en el campo de la evolución del software: los patrones evolutivos [Aoyama, 00] [Niertrasz, 00] [Amano, 00], el modelado dinámico [Belady, 76][Lehman, 00], la transformación de programas [Kozaczynsky, 92] [Berzins, 93], etc.

Nosotros al igual que otro autores [Banerjee, 87][Casais, 90][Mens, 01][Heckel, 01] [Wermenlinger, 01], consideramos la **evolución como un proceso de maduración**. El proceso de maduración comienza cuando empezamos a crear un sistema y está presente durante toda la vida del sistema. Concebimos un sistema como un conjunto de elementos en interacción y un modelo como una representación simbólica de éste. Según esta visión, un sistema software pasa, a lo largo de su desarrollo, por diferentes estadios de maduración. El cambio de un estadio a otro está motivado por la intervención del equipo de desarrollo para modificar el modelo del sistema.

8. EL MODELO MEDES

La visión de los conceptos de Teoría de Sistemas y de evolución que se aplican a lo largo de la presente tesis, tiene como marco epistemológico los trabajos realizados en torno a tales aspectos por el doctor Parets Llorca, especialmente su metodología MEDES [Parets, 95].

MEDES, es un **M**étodo de **E**specificación, **D**iseño y **E**volución del **S**oftware, cuya característica principal es que "trata de presentar el proceso de desarrollo de software como un proceso de modelado, realizado por un sistema de modelado, el cual asume la responsabilidad de representar la acción, la decisión y la finalidad propias del sistema modelado, considerando su carácter esencialmente evolutivo y adaptando como marco general de representación la Teoría de Sistemas General".

Desde el punto de vista de MEDES un sistema software consiste en un conjunto de sistemas que interactúan entre sí y con el entorno. El funcionamiento de un sistema viene determinado por su estructura, por tanto cualquier cambio en ésta produce también un cambio en su funcionamiento. El equipo de desarrollo es el responsable de ir madurando la estructura del sistema para hacer evolucionar su funcionalidad según crea

necesario. El equipo podrá realizar los cambios mediante un sistema software especial denominado meta-sistema.

De acuerdo a MEDES, un sistema software, representado en la figura 5, se compone de un conjunto de procesadores, esto es, entidades activas que realizan las acciones del sistema. Dentro de las acciones se distinguen dos tipos, las acciones funcionales que definen el comportamiento del sistema y las acciones estructurales o de evolución que permiten modificar la estructura del sistema y como consecuencia su comportamiento.

Para garantizar la consistencia en los cambios realizados, estas acciones tienen asociadas un conjunto de condiciones que determinan cuándo es posible ejecutar una acción y cuándo no. El módulo encargado de evaluar las condiciones asociadas a una acción y decidir si se lleva a cabo, recibe el nombre de sistema de decisión (SD). Las acciones finalmente realizadas en el sistema quedan registradas en forma de eventos dentro de la historia del sistema, estructural (HE) o funcional (HF), según corresponda.

Obviamente el sistema debe contar con una interfaz para comunicarse con el exterior. Es necesario diferenciar la interfaz de acción de la de evolución. La primera es el medio de comunicación con el entorno en lo que se refiere a la actividad para la que el sistema fue desarrollado. La segunda, es la vía a través de la que se producen las modificaciones estructurales en el sistema software y los mensajes llegan hasta ella procedentes del meta-sistema.

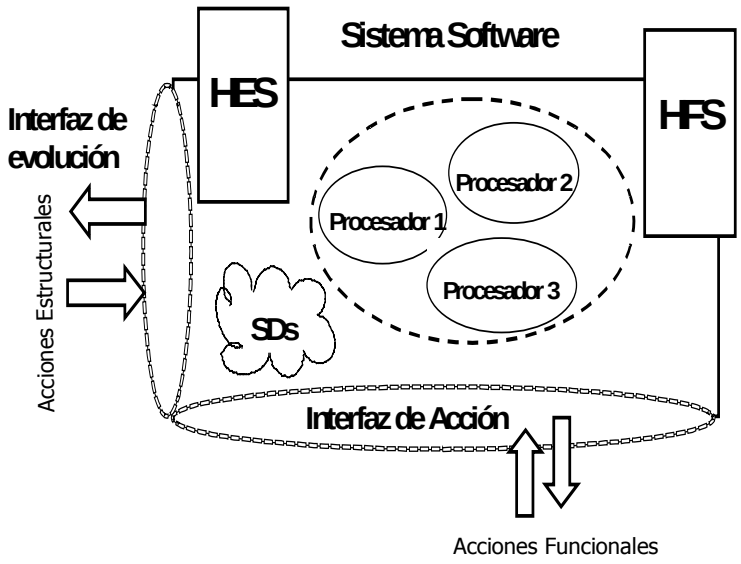


Figura 5: Sistema Software según MEDES

Para modificar la estructura del sistema, el equipo de desarrollo debe interaccionar con el meta-sistema. Podemos considerar el meta-sistema como un segundo nivel de abstracción, que permite al desarrollador la construcción y reconstrucción de los distintos componentes del sistema software del nivel inferior. No obstante, el meta-sistema no deja de ser también un sistema software, salvo que, en la mayoría de los casos, su estructura no cambia y su finalidad es siempre la misma.

Tal y como vemos en la figura 6, el autor se comunica con el meta-sistema haciendo uso de la interfaz de acción de éste. En consecuencia, el autor debe solicitar la ejecución de cualquier modificación sobre el sistema software, siempre a través del meta-sistema.

Éste actúa como intermediario y se encarga de enviar al sistema software la correspondiente acción estructural a través de la interfaz de evolución del sistema. Obviamente, dicha acción sólo será ejecutada si satisface las invariantes establecidas.

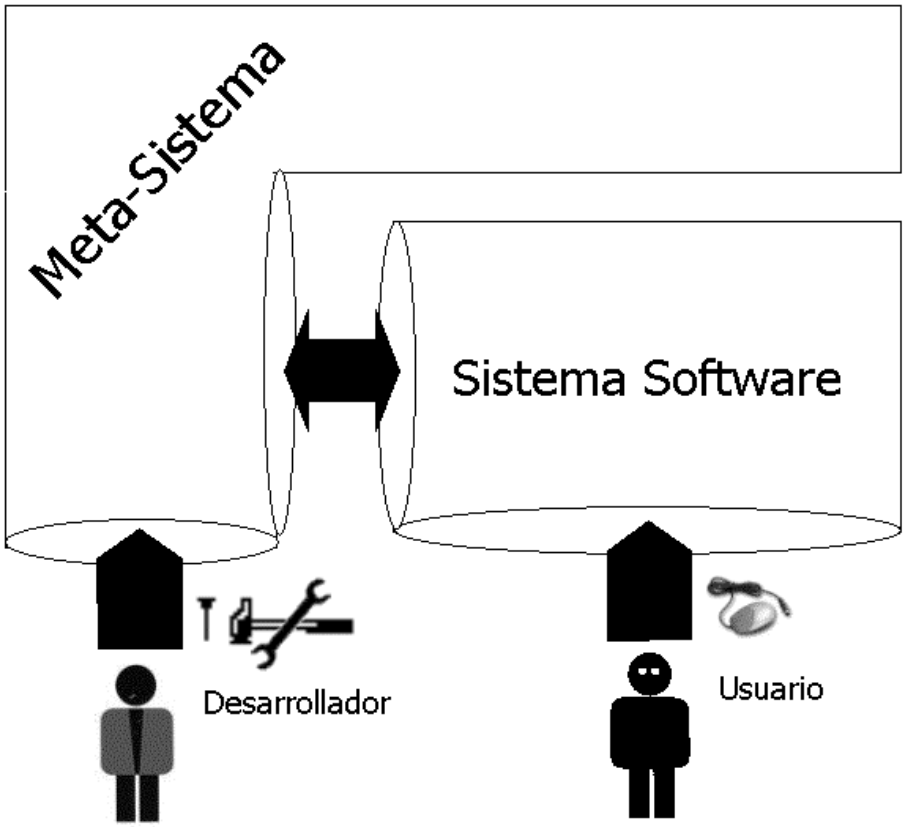
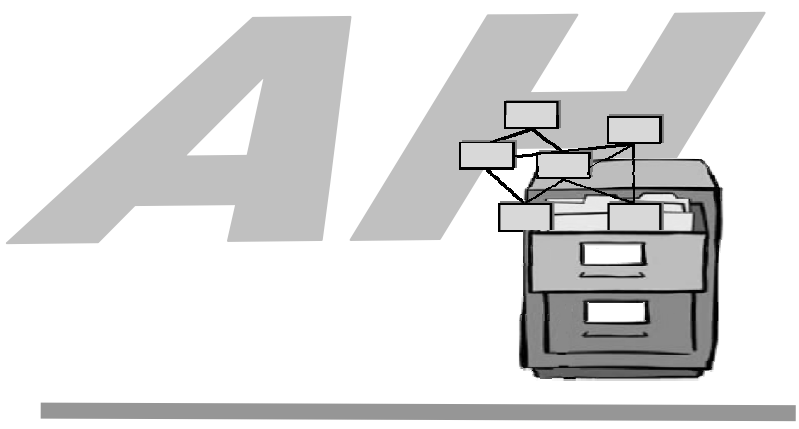


Figura 6: Dos niveles de abstracción en MEDES

CAPÍTULO 3

Sistemas Hipermedia Adaptativos



Resumen

En este capítulo se describe brevemente el origen de los sistemas hipertexto y sus sucesores, los sistemas hipermedia. A través de distintas definiciones encontradas en la literatura se presenta el concepto de hipermedia, describiendo los elementos que lo constituyen y resaltando sus principales ventajas. Se estudian los problemas más frecuentes durante la navegación de estos sistemas, así como las ayudas que se proporcionan para paliarlos. Se justifica la necesidad de adaptación al usuario y se introducen los sistemas hipermedia adaptativos, describiendo sus características y arquitectura generales. Finalmente se presenta y desarrolla una taxonomía que permite comparar los sistemas hipermedia adaptativos encontrados en la literatura científica respecto a ocho criterios diferentes. Tras esta revisión del estado del arte, se exponen las principales ventajas e inconvenientes encontradas en este tipo de sistemas.

Tabla de contenidos

Sistemas Hipermedia Adaptativos

1. Origen del Hipertexto

Los términos hipertexto e hipermedia son hoy en día muy comunes si hablamos de nuevas tecnologías de presentación y acceso a la información. Además se han revelando como una herramienta muy potente para la representación de documentos e información, así como para procesos formativos y educativos en todos los ámbitos.

Sin embargo, si buscamos los antecedentes de este enfoque de organización y acceso a la información debemos remontarnos a finales de los años 30. En tan temprana fecha, Vannevar Bush dejó escrito un borrador sobre un sistema de control, gestión y acceso a la documentación al que llamó MEMEX (*MEMory EXtender*).

Este manuscrito, escrito concretamente en 1939, no se publicó hasta 1945 con el título "As We May Think" [Bush, 45]. En él se exponía como idea principal la utilización de un principio de **asociación de conceptos entre recursos informativos**, de tal forma que el usuario pudiese acceder a estos recursos, independientemente de su tipo, simplemente utilizando la asociación de ideas.

Además se contemplaba la posibilidad de que estos enlaces se combinasen para formar rutas de viaje a través de los conceptos más importantes, anticipando incluso el nacimiento de una nueva profesión, conocida actualmente como *WebsRings*, esto es, personas que se encargan de establecer caminos útiles a través de ese gran volumen de información.

El sistema no llegó nunca se ser construido por la dificultad técnica que suponía. Hubo que esperar a la década de los 60 para que Doug Engelbart [Engelbart, 63], un investigador del *Stanford Research Institute*, dirigiese un proyecto de investigación destinado a desarrollar máquinas, basadas en sistemas informáticos, que permitiesen aumentar las capacidades y productividad humanas. Gracias al cual se introdujeron conceptos técnicos que han permitido gestionar gran cantidad de información según criterios jerárquicos y asociativos. Y sin los cuales no sería posible el hipertexto.

2. Concepto de Hipermedia

Como consecuencia de las numerosas investigaciones y proyectos que sucedieron a los pioneros, en la década de los 80 aparecieron una gran cantidad de aplicaciones y herramientas que facilitaban el desarrollo de sistemas hipertextuales. En los últimos años, estas herramientas han incorporado además capacidades multimedia, dando lugar de esta forma a los sistemas hipermedia.

El término **hipermedia** es el sucesor del término **hipertexto**, acuñado por Nelson [Nelson, 67] en 1965, y surge al combinar éste con el término **multimedia**. Son muchos los autores que han incluido en sus trabajos definiciones de hipermedia. La mayoría de ellos parten en su definición del término hipertexto. Balasubramanian [Balasubramanian, 93] define hipermedia como hipertexto con multimedia. McDaid

[McDaid, 91] considera el hipermedia como una extensión del hipertexto y resalta la posibilidad de navegar a través de material almacenado en muy diversos medios.

En general en todas las definiciones se resalta la posibilidad de realizar una lectura no-secuencial y la libertad del usuario para moverse a lo largo de los enlaces [Nelson, 87][Landow, 91][Conklin, 87]. Concretamente, Shneiderman [Shneiderman, 89] define hipermedia como "una base de datos que tiene referencias cruzadas activas, las cuales permiten al lector saltar, según sus deseos, a otras partes de la base de datos".

En todo sistema hipermedia (figura 1) es posible diferenciar dos elementos fundamentales: nodos y enlaces. Los **nodos** son las unidades de información ofrecidas por el sistema y pueden expresarse usando soportes informáticos muy diversos: texto, imágenes, audio, video, etc. Los **anclajes** permiten señalar las partes concretas del nodo de donde parte algún enlace. Aunque es relativamente sencillo insertar anclajes en puntos específicos del texto, con otros medios es más complejo debido a que no existe una forma estándar para su estructuración.

Los **enlaces** relacionan y establecen conexiones entre los nodos, lo cual permite al usuario navegar a través del sistema. Normalmente se añade al enlace una etiqueta que sirve para identificar el lugar de destino. En general, los enlaces son unidireccionales, aunque la implementación de enlaces bidireccionales no plantea ningún problema técnico.

Habitualmente el enlace se establece entre dos nodos distintos, pero también es posible hacerlo entre dos partes de un mismo nodo (enlace *span to span*). Los enlaces virtuales (el destino se determina dinámicamente a través del uso del documento), multidestino (al activarse el enlace se suministran varios destinos posibles), o con procedimientos adjuntos (que se ejecutan al seleccionar el enlace), incrementan las posibilidades del hipermedia.

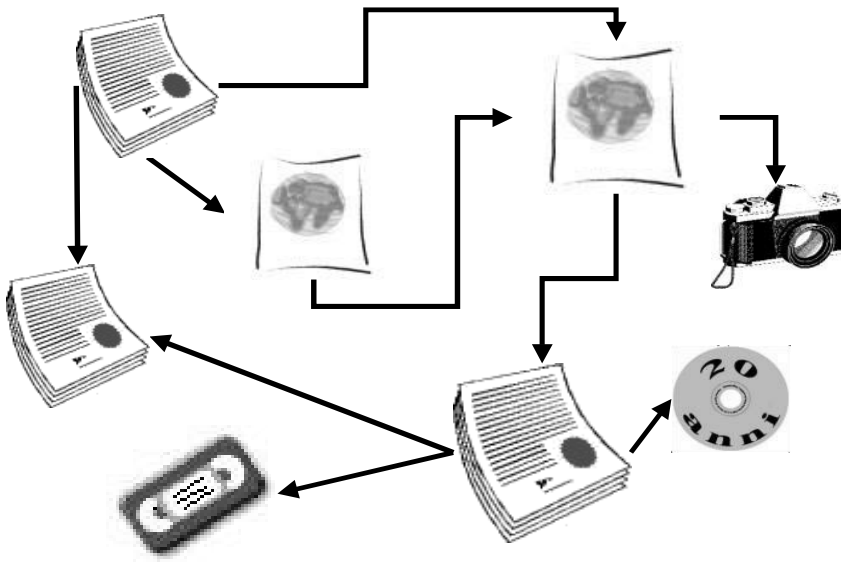


Figura 1: Sistema Hipermedia

El proceso de navegación es tal, que el material puede ser recorrido en un orden no lineal. De esta forma, dos documentos que contengan conceptos relacionados pueden ofrecer acceso directo de uno a otro, o a la porción de los mismos que sea pertinente.

Por otra parte, la estructura de enlaces posibilita varios caminos para acceder a la misma información. De este modo, dos usuarios pueden alcanzar un documento partiendo de un nodo distinto y realizando recorridos diferentes.

Es innegable, que este tipo de sistemas han sido ampliamente utilizados para comunicar conocimiento, especialmente en los últimos años debido a la aparición de la WWW (*World Wide Web*). La principal ventaja de un sistema hipermedia frente a un libro tradicional radica en la escritura y lectura no secuencial de sus documentos. Esto supone la existencia, y la adopción, del criterio de asociación de ideas y conceptos como principio organizador del conjunto de documentos. Obviamente, este tipo de organización no puede reflejarse adecuadamente en material impreso, ya que en éste predomina la linealidad.

3. NAVEGACIÓN HIPERMEDIA

Para que sea posible una existencia real de los conceptos de hipertexto e hipermedia, deben utilizarse aplicaciones que sean capaces de recorrer los vínculos entre los documentos. Este tipo de aplicaciones son conocidas como *browser*, navegador o visualizador, e integran al menos las siguientes funcionalidades:

- Ventanas de presentación de los documentos, las cuales suelen ser modificables en tamaño y posición.
- Dispositivos señaladores, que permiten la selección y acceso a los documentos mostrados en las ventanas.
- Enlaces dentro de los documentos, generalmente representados de forma distinta al resto del material informativo, que al ser pulsados, abren en una ventana el documento de destino.
- Posibilidad de ejecutar búsquedas en el texto completo que contienen los documentos, y/o búsquedas más rígidas utilizando lenguajes clásicos de consulta.

La visión que obtiene el usuario mediante el visualizador es transparente e integrada, de manera que no resulta complicado navegar de un documento a otro. Sin embargo, la gran cantidad de información proporcionada por el sistema implica que en ocasiones el usuario se encuentre perdido en el *hiperespacio* y a menudo tenga problemas para comprender la información mostrada en un nodo. Respecto a las dificultades que encuentra el usuario durante su recorrido por el sistema hipermedia, dos son los problemas más estudiados: desbordamiento cognitivo y desorientación [Nielsen, 90].

- **Desbordamiento cognitivo:**

El sistema hipermedia pretende facilitar una lectura exploratoria. Con lo cual, el usuario puede seguir los enlaces que considere oportunos, leyendo el material en cualquier orden. Además, si antes de llegar al final de un documento encuentra en éste un enlace interesante, el lector puede saltar hacia el nuevo documento, dejando pendiente la lectura de un trozo del documento actual.

Este proceso repetido varias veces requiere un esfuerzo y concentración importante, ya que hay que seguir muchos rastros al mismo tiempo. Puesto que cada esfuerzo adicional reduce los recursos mentales que se pueden dedicar a la comprensión, es razonable que

el lector tenga mayores dificultades para entender adecuadamente la información a la que accede.

- **Desorientación:**

El problema de desorientación consiste en que el lector pierde la noción de su situación dentro del hiperespacio, sin saber cómo ha llegado hasta allí. Esta sensación de pérdida se debe a la premura en la navegación de los usuarios, que cuando se sienten atraídos por un nodo, pasan rápidamente a leerlo, sin completar la lectura del nodo actual ni detenerse a reflexionar sobre el camino recorrido.

Una vez desorientados, reubicarse es una tarea difícil, ya que cada nodo tiene múltiples salidas distintas y no existe manera de saber si un enlace está cerca de un nodo visitado anteriormente.

Se han desarrollado varias **ayudas para la navegación** del usuario, que asisten a éstos en la tarea de encontrar la información que buscan, procurando evitar su desorientación. Algunas de estas ayudas son:

- Marcha atrás (*backtracking*). Los enlaces son bidireccionales para facilitar el regreso al nodo de origen.
- Chivatazo (*sneak preview*). Para cada enlace se escribe una breve descripción acerca del nodo de destino, que aparece al situarse sobre el anclaje sin necesidad de seleccionarlo.
- Enlaces resaltados (*highlighting links*). Se resaltan los enlaces previamente visitados para evitar la visita repetida a un nodo de forma no deseada.
- Anclajes únicos (*unique anchors*). Para evitar confusiones, no se permite emplear el mismo anclaje para apuntar a dos nodos distintos.
- Señaladores (*bookmarks*). Se permite al usuario crear marcas en un hiperdocumento para posteriormente localizar de forma rápida los elementos marcados.
- Lista histórica (*history list*). Se muestra la lista de nodos visitados hasta el momento, pudiendo el usuario repetir el camino (o parte de él) para reorientarse.
- Vistas globales (*birds-eye views*) o parciales (*fish-eye views*) que muestran la estructura del hiperespacio y la posición del usuario dentro de ella.

En los sistemas hipermedia no adaptativos, éstas y otras ayudas se realizan de forma idéntica para todos los usuarios. Nosotros consideramos que el principal problema de un sistema hipermedia, es que el usuario no se encuentre cómodo trabajando con él. Por lo tanto, la solución debe pasar por adaptar la estructura de enlaces y la presentación de la información contenida en los nodos a las características e intereses propias de cada usuario.

4. SISTEMA HIPERMEDIA ADAPTATIVO

Un **S**istema **H**ipermedia **A**daptativo (SHA) es un sistema hipermedia capaz de ajustar su presentación y navegación a las diferencias de los usuarios que lo utilizan, reduciendo así los problemas de desorientación y falta de comprensión propios de los sistemas

hipermedia no adaptativos. Surgen, a finales de los ochenta, con el propósito de mejorar la usabilidad de los sistemas hipermedia tradicionales. Y la mayoría de ellos realmente consiguen facilitar la actividad del usuario, haciendo que el sistema se adapte a determinadas características de éste.

El diseño de un sistema hipermedia adaptativo plantea cuatro cuestiones relativas a la adaptación [Medina, 02]:

- ¿Qué?, esto es, cuál es la funcionalidad del sistema susceptible de adaptación.
- ¿A qué?, a las características de quién o qué se ajusta el sistema.
- ¿Cómo?, es decir, qué técnicas y métodos utiliza el sistema para producir la adaptación.
- ¿Cuándo?, en qué momento durante el funcionamiento del sistema se produce la adaptación.

El autor construye los componentes del sistema hipermedia adaptativo en función de las decisiones de diseño que ha adoptado para resolver cada una de estas cuatro preguntas. Siguiendo a De Bra [DeBra, 00] existen tres elementos arquitectónicos que implícita o explícitamente están presentes en la mayoría de los sistemas hipermedia adaptativos:

- Modelo de dominio: Describe la estructura del dominio de la aplicación en términos de conceptos y relaciones entre conceptos.

- Modelo de usuario: Almacena las características del usuario que el sistema tiene en cuenta para realizar la adaptación. Suele incluir el conocimiento del usuario sobre los conceptos del modelo de dominio.

- Modelo de adaptación: Establece cómo la información del modelo de usuario influye en la adaptación del sistema. También especifica cómo y cuándo actualizar la información almacenada en el modelo de usuario.

Además, de acuerdo al mismo autor [DeBra, 99] un sistema hipermedia adaptativo ofrece tres funcionalidades básicas:

- Se registran las acciones que realiza el usuario cuando navega y sobre éstas se construye y mantiene un modelo de su conocimiento.

- El modelo del usuario se utiliza para clasificar los nodos y a partir de esta clasificación activar o desactivar los enlaces.

- Para asegurar que el contenido de una página es adecuado, el sistema oculta, resalta o reordena algunos fragmentos que son opcionales a la hora de presentar su información.

La figura 2 representa la arquitectura general de un sistema hipermedia adaptativo. En ella se muestra cuáles son sus componentes y cómo se relacionan entre sí. Además, asociada a cada componente, aparece una lista con las preguntas que el autor debe responder para diseñarla.

Esta arquitectura se ve claramente reflejada en la definición de Brusilovsky [Brusilovsky, 96], según la cual, existen tres criterios que todo sistema hipermedia adaptativo debe satisfacer:

a) ser un sistema hipermedia o hipertexto,

b) tener un modelo de usuario y

c) ser capaz de utilizar dicho modelo para adaptar el hipermedia.

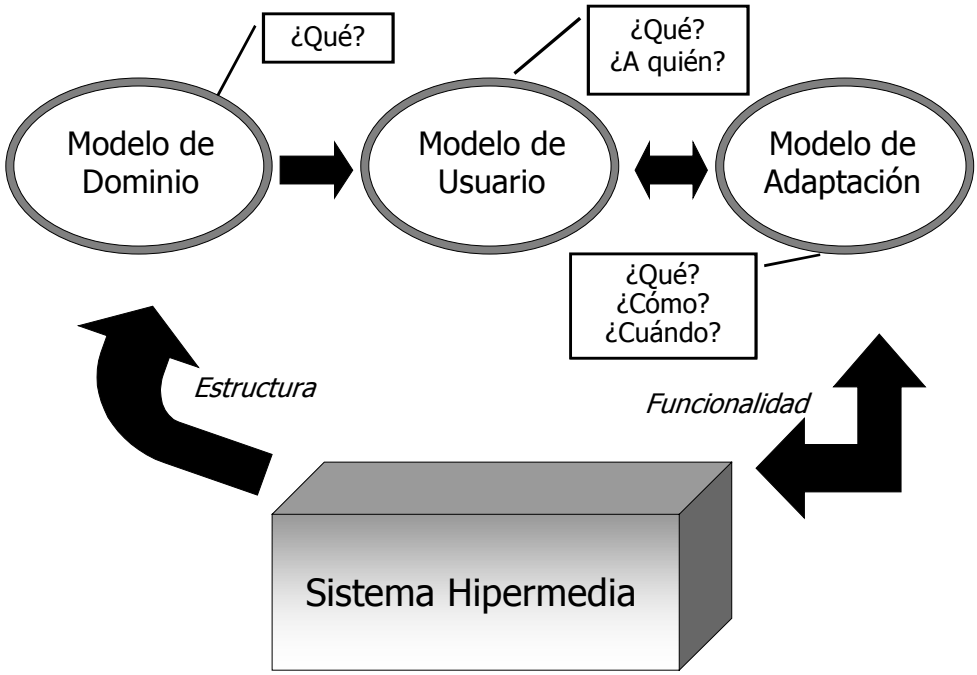


Figura 2: Arquitectura de un SHA

5. TAXONOMÍA DE SHA

A pesar de la relativa juventud de este campo de investigación, son muchos los trabajos realizados al respecto. Para contemplar el mayor número posible de ellos sin hacer demasiado extensa esta revisión, hemos desarrollado una taxonomía que nos va a servir para resumir las características principales de un sistema hipermedia adaptativo.

Dicha taxonomía ha sido publicada [Medina, 02c] en la segunda conferencia internacional sobre hipermedia adaptativo (AH'2002), y permite clasificar los sistemas hipermedia adaptativos encontrados en la literatura científica desde muy diversos puntos de vista.

En la tabla 1 se enuncian los ocho criterios utilizados en la taxonomía y se detallan las preguntas de diseño que se contemplan en cada uno de ellos. A continuación se expone el significado de cada uno, describiendo brevemente las peculiaridades de determinados sistemas respecto a dicho criterio. Por último, la tabla 4 recoge de forma esquemática la taxonomía desarrollada.

Tabla 1: Criterios de clasificación y preguntas de diseño

		¿QUÉ?	¿A QUÉ?	¿CÓMO?	¿CUÁNDO?
1	Aplicación del modelo	X			
2	Métodos de adaptación	X		X	
3	Objeto de la adaptación		X		
4	Tipo de prerrequisitos			X	
5	Integración de información	X			
6	Interacción usuario-adaptación			X	X
7	Creación de hiperdocumentos			X	X
8	Información contextual		X		

5.1 Aplicación del modelo

Cuando se diseña un modelo o se construye la correspondiente herramienta para el desarrollo de sistemas hipermedia adaptativos puede hacerse teniendo en mente un dominio de aplicación concreto o por el contrario pretender que la herramienta admita el desarrollo de sistemas hipermedia con diferentes dominios de información y/o campos de aplicación. Por lo tanto, la aplicabilidad del modelo permite la distinción de dos tipos de sistemas hipermedia adaptativos:

- Sistema hipermedia adaptativo general.
- Sistema hipermedia adaptativo específico.

En el primer tipo, los hiperdocumentos pueden tratar sobre temas muy diversos, mientras que en el segundo, todos los hiperdocumentos explicarán conceptos de un dominio de información uniforme. En la actualidad, la mayoría de los sistemas hipermedia adaptativos son específicos y persiguen fines didácticos o educativos.

Por ejemplo, ELM-ART [Brusilovsky, ELM] es un tutor de Lisp accesible a través de la Web que estructura su interfaz en cuatro zonas bien diferenciadas:

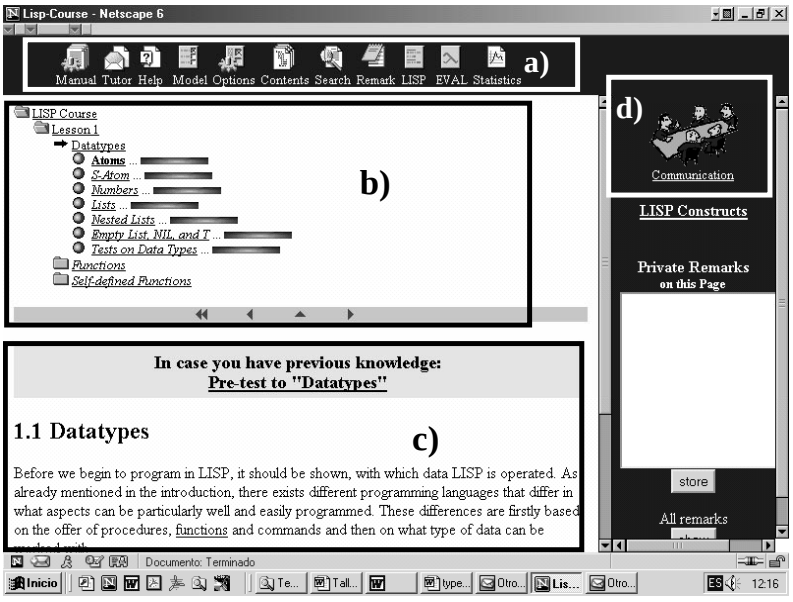


Figura 3: Interfaz gráfica de ELM-ART

a) Menú, donde es posible acceder a opciones de evaluación, ayuda, consultas vía e-mail a un tutor, estadísticas, contenidos del curso, comandos de Lisp, etc.

b) Estructura jerárquica de carpetas, mediante la cual el usuario puede navegar por las distintas lecciones del curso.

c) Contenido de la lección seleccionada.

d) Zona de discusión colectiva, que permite *chatear*, participar en foros, etc.

Otros ejemplos de sistemas específicos son el proyecto PUSH [Espinoza, 96] que organiza un manual con información sobre métodos de desarrollo de software o el proyecto ADAPTS [Brusilovsky, 99] que proporciona soporte técnico personalizado y diagnostico adaptativo para el mantenimiento de equipos complejos.

En contraposición, AHAM [DeBra, 98] es un modelo general que ha sido específicamente diseñado para desarrollar fácilmente herramientas autoras de hipermedia adaptativo. En relación a éste cabe destacar la arquitectura AHA [DeBra, 98b] [Wu, 00] que surgió inicialmente para servir a un curso adaptativo y a un kiosco virtual, pero que ha llegado a convertirse en un sistema hipermedia adaptativo genérico.

5.2 Métodos de adaptación

En cualquier sistema hipermedia dos aspectos son susceptibles de adaptación: la **estructura de enlaces** y la **información** contenida en las páginas. Así, los métodos de adaptación utilizados por los sistemas hipermedia adaptativos pueden clasificarse en dos grupos, según permitan adaptar la navegación o la presentación de la información ofrecida por el sistema.

Los métodos de adaptación utilizados más frecuentemente en los sistemas hipermedia adaptativos son resumidos en la tabla 2 [Brusilovsky, 96] [DeBra, 98c]. En la tabla 3 se exponen las técnicas de adaptación utilizadas a más bajo nivel para implementar estos y otros métodos.

Tabla 2: Métodos de adaptación más utilizados

Adaptación de la Presentación (métodos)	
Explicación adicional	Proporcionar información adicional (ejemplos, ilustraciones, comentarios, etc.) a aquellos usuarios que la necesiten.
Explicación de prerrequisito	Incluir información sin la que el usuario no comprendería el resto de la página. Permite compensar la falta de un conocimiento requerido.
Explicación comparativa	Incluir información sobre otros conceptos conocidos por el usuario que están relacionados con el concepto descrito en la página actual.
Variantes	La misma información es presentada a cada usuario con distinto nivel de especialización, idioma, grado de verborrea, etc.
Ordenación	La información de la página es ordenada de acuerdo a algún criterio, que dependerá de las características del usuario.
Adaptación de la Navegación (métodos)	
Consejo global	Sugerir un camino de navegación global: un conjunto de páginas y el orden de lectura.
Consejo local	Sugerir la siguiente página a visitar.

Soporte global de orientación	Mostrar una vista de la estructura de enlaces completa y la posición actual del usuario, indicando partes visitadas, deseables y prohibidas.
Soporte local de orientación	Mostrar una parte de la estructura de enlaces, normalmente uno o dos niveles arriba o debajo de la página actual.
Vistas personalizadas	Vista de la estructura de enlaces orientada a la meta del usuario.

Tabla 3: Técnicas de adaptación más utilizadas

Adaptación de la Presentación (técnicas)	
Texto condicional	Toda la información sobre un concepto se divide en trozos y a cada trozo se le añade una condición. Cuando se visualiza una página sobre un concepto sólo aparecen los trozos cuya condición sea cierta.
Estirar / Encoger texto	Los fragmentos se pueden expandir o encoger a gusto del usuario, pero el sistema determina el estado inicial de los fragmentos en la presentación de la página.
Variantes de fragmento	Para seleccionar la variante de un fragmento que se visualiza es necesario consultar el modelo de usuario.
Variantes de pagina	El autor prepara variantes de la página completa, y luego con el modelo de usuario se decide que variante se presenta. La desventaja de esta técnica frente a la anterior es que las páginas suelen solaparse.
Técnica basada en marcos	Se seleccionan los fragmentos que se van a utilizar para montar la página y el orden de aparición de los mismos. Los fragmentos pueden ser frases, e incluso palabras, y en ocasiones habrá que utilizar técnicas de lenguaje natural para que el resultado no suene artificial.
Adaptación de la Navegación (técnicas)	
Consejo directo	Botón <siguiente> que lleva al próximo mejor nodo que el usuario debe visitar.
Ordenación de enlaces	Ordena todos los enlaces de una página de acuerdo al modelo de usuario (especialmente a la meta de éste).
Ocultación de enlaces	Se ocultan los enlaces a las páginas no relevantes. Se cambia el color del texto del anclaje al color del texto que lo rodea. Intenta reducir la sobrecarga cognitiva.
Anotación de enlaces	Se aumenta el enlace con algún comentario (textual o visual) que nos indica la relevancia del enlace, si el usuario ya conoce el concepto descrito por la página enlazada o si la página a la que lleva el enlace es deseable para el usuario (es capaz de entenderla).
Borrado de enlaces	Se borran los anclajes de los enlaces no deseables. Esta técnica trabaja bien en enlaces no-contextuales, ya que no es conveniente borrar el anclaje si es una palabra integrada en un párrafo (en este caso es mejor la ocultación).
Deshabilitación de enlaces	Se elimina la funcionalidad del enlace. Conviene utilizar esta técnica junto con la de ocultación o la de anotación para que el usuario no espere que el enlace funcione.

La técnica de texto condicional, la de estirar/encoger texto o la de variantes de fragmento, permite implementar los métodos de explicación adicional, explicación de prerrequisito y explicación comparativa. La técnica basada en marcos implementa el

método de ordenación. La técnica de consejo directo es utilizada para implementar el método de consejo local. Y la técnica de ordenación de enlaces permite llevar a la práctica el método de consejo global.

Aunque dos sistemas utilicen el mismo método de adaptación, la forma de implementarlo puede variar sustancialmente, incluso utilizando la misma técnica.

Por ejemplo, en AHA [DeBra, 98b] se utiliza la anotación de enlaces mediante colores y se permite al usuario establecer sus preferencias respecto al esquema de coloreo. En AHM [DaSilva, 97] la anotación se realiza a través de un comentario que indica el estado actual del nodo al que apunta el enlace. Mientras que en ADAPTS [Brusilovsky, 99] la anotación puede ser tanto visual (colores o iconos) como textual (en forma de comentarios).

5.3 Objeto de la adaptación

Autores como Alatalo y Peräaho [Alatalo, 01] defienden un concepto muy general de adaptación, donde además de la adaptación al usuario caben otros casos de adaptación como, por ejemplo, la adaptación a los distintos tipos de aparatos en tecnología móvil.

No obstante, lo cierto es que la mayoría de los trabajos realizados se centran en el usuario, modelando sus características, preferencias y deseos para producir la adaptación en el sistema. Dentro de éstos, podemos diferenciar aquellos en las que se tiene un **modelo de usuario** de aquellos que manejan un **modelo de grupo de usuarios**.

5.3.1 Modelo de usuario

El modelo de usuario es una representación interna del usuario almacenada, mantenida y consultada por el sistema, con el fin de adaptarse a cada usuario, proporcionándole la navegación y presentación de la información que necesita. El estado actual del usuario se representa a través de sus intereses, conocimientos, preferencias y experiencias previas (tanto en la materia como en el uso de sistemas hipermedia).

Mientras que la existencia de un modelo de usuario es común a la mayoría de los sistemas, la forma en que éste es representado varía sustancialmente, como podemos comprobar en los dos ejemplos presentados a continuación.

En el modelo AHAM [DeBra, 98] para cada usuario se mantiene una estructura tabular, en la que se asocia a cada concepto del modelo de dominio una lista de parejas (atributo, valor). A cada concepto se le asocian atributos del tipo: nivel de conocimiento, probabilidad de ser la meta actual del usuario, si el usuario ha leído algo sobre él, si el usuario está preparado para entenderlo, etc. Además se “abusa” de los conceptos para introducir aspectos como experiencia, preferencias, etc.

En el proyecto KBS Hyperbook [Henze, 99] el conocimiento del usuario es modelado mediante un vector, donde cada componente es una probabilidad condicional que indica la estimación del conocimiento que tiene el usuario sobre un tema. Una red bayesiana implementa el modelo de usuario, actualizando el vector de conocimiento a partir de las observaciones realizadas sobre el trabajo del mismo.

5.3.2 Modelo de grupo de usuarios

Los sistemas que gestionan un modelo de grupo de usuarios, tienen en cuenta para llevar a cabo la adaptación un conjunto de individuos en lugar de un único individuo. Sin embargo, las **recomendaciones** realizadas por el sistema pueden ser **personalizadas** (a un individuo) **o colectivas** (al conjunto de individuos). Las formas en que el modelo de grupo puede ser representado son también muy diversas. A continuación se presentan dos ejemplos que ilustran esta diversidad.

El sistema propuesto por Bollen [Bollen, 98][Bollen, 00] es capaz de reestructurar su red hipertexto mientras que está siendo recorrida, para que ésta converja hacia la representación de los modelos mentales que el grupo de usuarios tiene sobre el contenido del sistema. De esta forma se utiliza el conocimiento del grupo para facilitar la navegación de un individuo.

Por su parte, INTRIGUE [Ardissono, 01] [Ardissono, 01b] es un sistema capaz de hacer recomendaciones en el ámbito de los viajes organizados. Un modelo de grupo es explotado para poder mantener de forma separada las preferencias de subgrupos heterogéneos y combinarlas para identificar soluciones satisfactorias al grupo completo. Cada subgrupo modela un tipo de usuario, es decir, un conjunto de personas con preferencias y características similares.

5.4 Tipo de prerrequisitos

En los sistemas hipermedia adaptativos, los prerrequisitos se utilizan para establecer bajo qué circunstancias (normalmente el grado de conocimiento o las páginas visitadas con anterioridad) el usuario podrá visitar una determinada página. De este modo, los prerrequisitos describen los múltiples caminos que los usuarios pueden seguir en el sistema hipermedia.

Según Hübscher [Hübscher, 01] se llama prerrequisito a dos conceptos diferentes y confundirlos puede conducir a un diseño defectuoso.

- Prerrequisitos como **mecanismo de ordenación**: Establecen parcialmente el orden en que las páginas pueden visitarse. Si la página Pa es un prerrequisito de la página Pb (Pa $\rightarrow$ Pb) entonces Pa debe visitarse antes que Pb.

- Prerrequisitos **pedagógicos**: Establecen la relación entre dos conceptos con respecto al aprendizaje. Si el concepto A es requerido para comprender el concepto B entonces A es un prerrequisito pedagógico de B (A $\Rightarrow$ B).

A menudo los prerrequisitos pedagógicos pueden transformarse de forma directa en prerrequisitos de ordenación.

A $\Rightarrow$ B es transformado en Pa $\rightarrow$ Pb, donde la página Pa explica el concepto A y la página Pb explica el concepto B.

Sin embargo, existen situaciones en las que no es así. Por ejemplo, ciertas metodologías de enseñanza requieren que los estudiantes descubran por sí mismos la necesidad de aprender unos conceptos para comprender otros.

Siguiendo esta metodología, el prerrequisito pedagógico A $\Rightarrow$ B es transformado en el prerrequisito de orden Pb $\rightarrow$ Pa.

5.5 Capacidad de integración de información

Según la definición de Henze en [Henze, 01] si nos fijamos en la capacidad del sistema para incorporar nueva información, diremos que un sistema hipermedia adaptativo es **abierto** cuando permite tratar todas las unidades de información del mismo modo, independientemente de cuál sea su origen. Así, los sistemas abiertos contribuyen a un acceso universal, permitiendo adaptar la información localizada en cualquier sitio de Internet a las necesidades particulares de cada usuario.

La forma concreta en que se logra la capacidad de apertura varía notablemente de un sistema a otro.

Por ejemplo, en el proyecto Hyperbook KBS [Henze, 99] se sigue un enfoque de indexación, donde cada recurso de información incorporado en el hiperlibro es indexado mediante un conjunto de ítems de conocimiento que describen el contenido del recurso (no su origen).

5.6 Interacción del usuario con la adaptación

Ya comentamos en la sección 5.3 que, en la mayoría de los casos, los usuarios de un sistema hipermedia adaptativo son los que se benefician de la adaptación realizada por éste, ya sea de forma individual o en grupo. Sin embargo, hasta ahora no hemos analizado el papel que puede jugar el usuario en este proceso de adaptación. Fundamentalmente, la interacción del usuario con la adaptación puede abarcar dos características:

a) Gestión del modelo de usuario: creación y actualización.

b) Control sobre las adaptaciones realizadas en el sistema.

En función de si el usuario interviene directa o indirectamente en la **gestión del modelo** de usuario (a), distinguimos dos tipos de sistemas:

- Sistema hipermedia **adaptable**: El usuario establece explícitamente sus preferencias o proporciona su perfil a través de un formulario. El modelo de usuario es únicamente actualizado si el usuario lo solicita de forma explícita.

- Sistema hipermedia **adaptativo**: El modelo de usuario se construye observando la navegación del usuario y es automáticamente actualizado a medida que el usuario recorre la información.

La mayoría de los sistemas hipermedias adaptativos son también adaptables, porque necesitan un modo de inicializar el modelo de usuario o porque permiten a los usuarios ajustar explícitamente el modelo de usuario.

Un ejemplo de sistema adaptable y adaptativo es el enfoque *Intensional Hypertext* propuesto en [Wadge, 01]. Este sistema mantiene un modelo de usuario implícito y presenta de forma más explícita el modelo de dominio, permitiendo que los usuarios

determinen en todo momento qué versión de un documento desean ver. Sin embargo, para que los usuarios puedan, si lo desean, beneficiarse de las ventajas de un modelo de usuario automático, también soporta la selección automática de versión e incluso una combinación de ambas.

Respecto al **control sobre la adaptación** que el sistema permite al usuario (b), distinguimos dos tipos de sistemas, según:

- El usuario no tenga control explícito sobre las adaptaciones realizadas.
- El usuario pueda, si lo desea, tener algún control sobre el comportamiento adaptativo del sistema.

Los seguidores de esta segunda alternativa creen que "poner en manos del usuario cierto control sobre el sistema incrementa su confianza en el mismo". Para ello, los sistemas deben ser altamente interactivos, permitiendo al usuario alterar la salida y controlar la adaptación del sistema.

Dentro de este enfoque, podemos ubicar el proyecto PUSH [Espinoza, 96], en el que los usuarios tienen control sobre la adaptación, sin llegar a ser un control continuo que les suponga un gasto de tiempo excesivo.

El sistema PUSH realiza la adaptación teniendo en cuenta la tarea que efectúa en cada momento el usuario. El usuario informa al sistema de su tarea inicial y posteriormente el sistema ejecuta un plan de inferencia para estimar automáticamente la tarea actual del usuario. Para reducir los errores, el sistema permite al usuario modificar la tarea inferida siempre que lo considere oportuno.

5.7 Creación de hiperdocumentos

Considerando el momento en que los documentos de un sistema hipermedia son creados, un sistema puede ser **estático** o **dinámico** según los documentos existan de forma previa o sean generados en el momento en que se solicitan. El concepto de hipermedia dinámico no debe confundirse con el concepto de hipermedia adaptativo, ya que como muestran sus correspondientes definiciones son dos cosas muy distintas:

- El hipermedia adaptativo existe previamente a su uso, sólo que su contenido y estructura de enlaces es adaptado en función de un modelo de usuario, para ajustarse a las metas, preferencias y características de cada usuario particular.
- El hipermedia dinámico no existe hasta el momento en el que el usuario lo explora, siendo dinámicamente creado mediante el empleo de técnicas automáticas de generación de texto.

El enfoque del Macronodo propuesto por Elena Not y Maximmo Zancanaro [Not, 99][Not, 00] pretende mediar entre estos dos conceptos y conseguir un sistema hipermedia dinámico y adaptativo, que permita la generación de páginas bajo demanda (propia del hipermedia dinámico), pero también una reutilización substancial de la información existente (propia del hipermedia adaptativo).

Para ello, este enfoque reduce la granularidad de la adaptación a nivel de macronodo (trozo de información) y construye los nodos del sistema hipermedia componiendo diferentes macronodos obtenidos de un repositorio central donde éstos se encuentran almacenados y convenientemente anotados. Para concatenar los macronodos se utilizan estrategias de discurso y reglas lingüísticas tomadas prestadas del campo de la generación de lenguaje natural

5.8 Información contextual

A veces para refinar el proceso de adaptación, el sistema tiene en cuenta información de contexto. De este modo, el sistema hipermedia adaptativo considera no sólo las características del usuario sino también su entorno.

El contexto es algo que depende de la tarea que se realiza y de las variables disponibles para modelar dicha tarea. En los sistemas hipermedia, la principal tarea es la navegación del usuario y las entidades involucradas: el usuario y los documentos disponibles. En consecuencia, los distintos tipos de información contextual que puede ser manejada por el sistema a la hora de producir la adaptación son los siguientes:

- **Contexto del usuario**: Incluye información tal como el papel del usuario en una organización o grupo, y su localización física.

- **Contexto del documento**: Contenido, formato, fecha de creación, propósito, servidor en el que reside, velocidad de descarga, etc.

- **Contexto espacial del documento**: Representa la localización del documento dentro del dominio de información en el que se encuentra inmerso. Cuando un usuario accede a un documento lo hace seleccionando uno de todos los posibles caminos que llegan hasta él. Por lo tanto, parece sensato adaptar el documento teniendo en cuenta el camino que se ha seguido para alcanzarlo.

El enfoque propuesto por los autores C. Bailey, S. El-Beltagy y W. Hall [Bailey, 01] utiliza el contexto espacial del documento para aplicar de forma coherente la técnica de aumentación de enlaces, que consiste en sustituir ciertas palabras del documento por enlaces a páginas relacionadas.

Para la aplicación de esta técnica de adaptación, la información contextual es esencial, ya que conociendo el contexto del documento es posible determinar el significado correcto de una palabra del documento cuando ésta tiene varias acepciones, evitando así enlaces a páginas irrelevantes para el contexto actual.

5.9 La taxonomía

A continuación, en la tabla 4, se resume cada uno de los criterios de la taxonomía, indicando en cada caso las distintas posibilidades que encontramos respecto a él en la literatura científica.

Tabla 4: Taxonomía de SHA

<table>
<tr><th>CRITERIO</th><th colspan="4">TIPOS</th></tr>
<tr><td rowspan="2">Dominio de Aplicación</td><td colspan="4">Sistema hipermedia adaptativo general: Los documentos tratan sobre temas muy diferentes.</td></tr>
<tr><td colspan="4">Sistema hipermedia adaptativo especifico: Todos los documentos explican conceptos de un dominio de información uniforme.</td></tr>
<tr><td rowspan="5">Adaptación a</td><td rowspan="3">Usuario</td><td rowspan="3">Representación del Modelo de Usuario</td><td>Parejas (atributo/valor).</td></tr>
<tr><td>Modelo bayesiano.</td></tr>
<tr><td>Otros.</td></tr>
<tr><td colspan="2">Grupo de usuarios</td><td>Recomendaciones personalizadas.</td></tr>
<tr><td colspan="3">Recomendaciones a un grupo de usuarios.</td></tr>
<tr><td colspan="4">Otros. Por ejemplo, adaptación a diferentes tipos de aparatos en tecnología de móviles.</td></tr>
<tr><td rowspan="6">Métodos Adaptativos</td><td colspan="2" rowspan="3">Navegación Adaptativa</td><td>Consejo</td><td>Local o Global.</td></tr>
<tr><td>Soporte de orientación</td><td>Local o Global.</td></tr>
<tr><td colspan="2">Vistas personalizadas.</td></tr>
<tr><td colspan="2" rowspan="3">Presentación Adaptativa</td><td colspan="2">Explicación adicional / prerrequisito / comparativa.</td></tr>
<tr><td colspan="2">Variantes de una misma explicación.</td></tr>
<tr><td colspan="2">Ordenación.</td></tr>
<tr><td rowspan="2">Tipo de Prerrequisitos</td><td colspan="4">Prerrequisitos pedagógicos: Las relaciones entre páginas obedecen a aspectos relacionados con el aprendizaje.</td></tr>
<tr><td colspan="4">Prerrequisitos como mecanismo de ordenación: Tratan de establecer un orden parcial entre las páginas.</td></tr>
<tr><td rowspan="2">Capacidad para Integrar Información</td><td colspan="4">Sistema hipermedia adaptativo abierto: Estos sistemas pueden integrar recursos de información localizados en cualquier sitio de la WWW.</td></tr>
<tr><td colspan="4">Sistema hipermedia adaptativo cerrado.</td></tr>
<tr><td rowspan="3">Interacción del Usuario con la Adaptación</td><td colspan="4">Sistema hipermedia adaptable: El modelo de usuario se actualiza sólo después de una solicitud explícita del usuario.</td></tr>
<tr><td colspan="2">Sistema hipermedia adaptativo: El modelo de usuario se actualiza automáticamente mientras el usuario navega.</td><td colspan="2">El usuario, si lo desea, puede tener algún control sobre el comportamiento adaptativo.

El usuario no tiene control explícito sobre la adaptación.</td></tr>
<tr><td colspan="4">Sistema hipermedia adaptable/adaptativo.</td></tr>
<tr><td rowspan="2">Creación de Documentos</td><td colspan="4">SHA dinámico: Los documentos se crean dinámicamente bajo demanda.</td></tr>
<tr><td colspan="4">SHA no dinámico: Los documentos existen de forma previa a su uso, aunque su presentación sea adaptada a cada usuario.</td></tr>
<tr><td rowspan="3">Información Contextual</td><td colspan="4">Contexto de usuario: Papel de usuario en un grupo, localización física, etc</td></tr>
<tr><td colspan="4">Contexto textual: Frase, párrafo o documento donde está inmerso el texto.</td></tr>
<tr><td colspan="4">Contexto espacial: Camino de navegación seguido por el usuario hasta llegar a la página actual.</td></tr>
</table>

6. VENTAJAS E INCONVENIENTES DE LOS SHA

A partir del proceso de revisión y análisis realizado sobre la situación actual de los sistemas hipermedia adaptativos [Medina, 02][Medina, 02e], observamos que debido a sus características adaptativas, estos sistemas ofrecen a los usuarios una serie de beneficios que los convierten en herramientas muy potentes.

Para empezar, los sistemas hipermedia adaptativos heredan las **ventajas** propias de los sistemas hipermedia tradicionales, esto es:

- la *pluralidad de medios* y los *aspectos visuales* hacen más atractivo el proceso de apredizaje,

- al mismo tiempo, que la *lectura* de la información en un orden *no-secuencial* proporciona al usuario mayor libertad de navegación.

Además de estas cualidades, existe un importante número de beneficios derivados de su capacidad de adaptación.

- En primer lugar, este tipo de sistemas inducen al autor a *estructurar mejor* su *conocimiento*, lo que le facilita el desarrollo del sistema y su posterior mantenimiento.

- También, es muy común que se establezcan prerrequisitos entre los documentos del sistema, incitando u obligando al lector a seguir un *orden coherente durante el recorrido de la información.*

- Esta posibilidad, unida a la aplicación de técnicas para personalizar la presentación de la información contenida en el documento, permite reducir sustancialmente los problemas de comprensión. Obviamente, los usuarios encuentran *menor dificultad para comprender los conceptos* cuando encuentran la información de las páginas ajustada a sus características e intereses.

- Los sistemas que, además, tienen en cuenta las preferencias y gustos del usuario *incrementan el grado de satisfacción* de quienes los usan.

- Igualmente es muy común la aplicación de métodos de soporte a la orientación, que *disminuyen los problemas de desorientación* y pérdidas en el hiperespacio, utilizando técnicas como, por ejemplo, la anotación de enlaces.

- Es más fácil para los usuarios de estos sistemas obtener una *vista de conjunto de la estructura de enlaces* y su posición en ella.

- Asimismo, debido a la utilización de técnicas adaptativas como la anotación y ocultación de enlaces, se reduce el número de veces que el usuario sigue, de forma no premeditada, un enlace hacia un documento anteriormente visitado.

- Por último, también, se puede *dirigir al usuario* a través del hiperespacio, *sin que pierda totalmente la libertad* de navegación, aplicando métodos adaptativos como el consejo local o el consejo global.

Como conclusión podemos resaltar el hecho de que los sistemas hipermedia adaptativos permiten atender las necesidades especiales de algunos usuarios, esto es, son *sistemas solidarios*. Sin embargo, a pesar de sus muchas cualidades, también surgen algunos **inconvenientes** durante el uso de los sistemas hipermedia adaptativos, por ejemplo:

- Los *cambios* en la estructura de enlaces y en el contenido de las páginas *no son generados bajo demanda* del usuario, por lo que en ocasiones pueden desorientarlo. Situación que se agrava si el usuario no tiene acceso a la información que acerca de él maneja el sistema para producir la adaptación.

- Además, dependiendo del uso que el lector desea dar a un concepto puede no tener sentido que el sistema le obligue a leer todos los prerrequisitos de dicho concepto.

- Por otro lado, normalmente los sistemas hipermedia adaptativos son *privados, no están distribuidos* y *no se han desarrollado para Internet*, lo cual limita su acceso.

Del mismo modo, la construcción del sistema hipermedia adaptativo plantea una serie de problemas desde la perspectiva del autor:

- Es obvio que *se complica la tarea de los autores*, aunque esta desventaja puede suavizarse si se les facilita una herramienta de autor apropiada.

- Además, *las técnicas de adaptación actuales están orientadas al texto* y en escaso grado se pueden aplicar sobre otros medios como audio, vídeo, imagen, etc.

- Pero aún más preocupante es el hecho de que los procesos de diseño, construcción y mantenimiento (el ciclo de vida completo) de los sistemas hipermedia adaptativos no estén suficientemente considerados. En particular, las *herramientas autoras no incorporan mecanismos que faciliten los cambios en el sistema*, durante y después de su construcción.

CAPÍTULO 4

Sistemas Educativos

Resumen

La educación es un área de aplicación idónea para los sistemas hipermedia adaptativos. En el presente capítulo se hace una breve revisión histórica de los sistemas educativos. Resaltando la importancia en este campo de los sistemas tutores inteligentes. Se describen las principales características de este tipo de sistemas, centrándonos en su arquitectura y en el uso de diversas técnicas de apoyo educacional. También se establece la necesidad de colaboración entre los sistemas tutores inteligentes y los sistemas hipermedia adaptativos, para desarrollar sistemas educacionales inteligentes y adaptativos que combinen los beneficios de ambos.

Tabla de contenidos

Sistemas Educativos

1. INTRODUCCIÓN

La educación ha sido siempre el área de aplicación más importante de los sistemas hipermedia adaptativos. De hecho, gran número de los métodos y técnicas de adaptación al usuario fueron expresamente desarrollados para sistemas educativos.

Además, la mayoría de estas investigaciones estuvieron inspiradas en el campo de los ITSs (*Intelligent Tutorial Systems*). Según [Brusilovsky, 00], fue precisamente, la combinación de un sistema tutor inteligente y el material organizado como un hipermedia, el punto de partida natural para la investigación en el desarrollo de sistemas hipermedia adaptativos con fines educativos.

La introducción de la Web ha influido, no sólo en el número de sistemas hipermedia adaptativos para la educación, sino también en la forma en que éstos están siendo desarrollados:

- Mientras que los primeros sistemas eran esencialmente sistemas de laboratorio construidos para explorar algún nuevo método de adaptación y su utilidad en un contexto educacional,

- en la actualidad existen sistemas que proporcionan un entorno completo, incluso herramientas de autor, para el desarrollo de cursos basados en web.

La aparición de estos sistemas refleja la madurez de los sistemas hipermedia educativos, pero también, responde a la demanda que existe en la Web de cursos formativos a distancia, capaces de personalizarse a cada usuario.

2. SISTEMAS EDUCATIVOS INTELIGENTES Y ADAPTATIVOS

La educación basada en web, WBE (*Web-based education*), es hoy día un punto fuerte de investigación y desarrollo. Según algunos autores, por ejemplo [Iglesias, 03], esto se debe fundamentalmente a dos de sus ventajas:

a) Independencia de la clase.

b) Independencia de la plataforma.

Tradicionalmente los cursos web eran un conjunto de documentos hipertexto estáticos que no se ajustaban a sus usuarios. Sin embargo, desde finales de los 90, varios equipos de trabajo están implementando sistemas inteligentes y adaptativos para educación en la Web [Brusilovsky, 99b].

Los AIES (*Adaptive and Intelligent Educational Systems*) proporcionan inteligencia y adaptación al usuario heredando las características de los sistemas tutores inteligentes y los sistemas hipermedia adaptativos. Los primeros son sistemas instructivos asistidos por ordenador que especifican qué enseñar y cómo enseñar [Wenger, 87]. Los segundos,

revisados en el capítulo anterior, permiten adaptar la estructura y navegación del curso web al usuario que lo realiza.

3. SISTEMAS TUTORES INTELIGENTES

Debido a la estrecha relación que existe entre los sistemas hipermedia adaptativos y los sistemas tutores inteligentes, la presente sección se dedica a describir las características principales de éstos últimos, con la intención de disponer de información suficiente para detectar posibles puntos de colaboración entre ambos.

3.1 Antecedentes e historia

Los sistemas de enseñanza desarrollados anteriormente a los sistemas tutores inteligentes se conocen con el nombre de *Computer Assisted Instruction*, CAI de forma abreviada.

Se trataba de cursos, normalmente muy extensos, donde el diseño e implementación del sistema se realizaba a medida, presentando entre otros los siguiente problemas:

- Poca comunicación alumno-profesor.
- Modelos establecidos con independencia de las características del alumno y sus preferencias.
- Poco flexible ante cambios.

La evolución temporal de los sistemas CAI [Urretavizcaya, 01] se muestra en la siguiente figura.

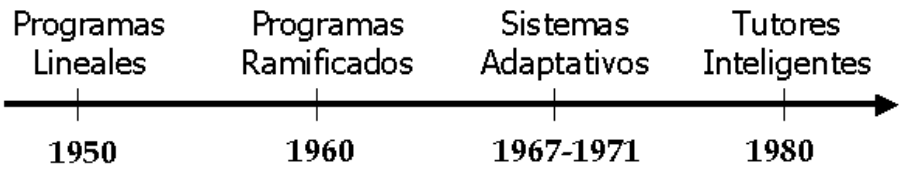


Figura 1: Antecedentes de los ITSs

Los **programas lineales** muestran el conocimiento de una manera lineal, en consecuencia, no es posible una reorganización del programa de enseñanza. Los **programas ramificados** tienen un número fijo de temas igual que los lineales, pero pueden cambiar su manera de actuar basándose en las respuestas del alumno. El objetivo de los **sistemas adaptativos** es ajustar el proceso de enseñanza al alumno, siendo capaces de generar un problema acorde con el nivel del alumno, construir la solución y diagnosticar la respuesta del alumno.

Los **tutores inteligentes** mejoran el proceso de enseñanza aplicando diversas técnicas de Inteligencia Artificial que facilitan y hacen más agradable y efectivo el aprendizaje del alumno. En estos sistemas, el dominio de conocimiento está acotado y bien estructurado. Mantienen un modelo del alumno para dirigir y adaptar la enseñanza. Y mejoran la comunicación tutor-alumno, permitiendo que el alumno realice preguntas al tutor.

Como puede observase en la figura 1, la idea de aprovechar herramientas informáticas en la enseñanza [García, 95][García, 96] se remonta a la década de los 50. No obstante, tienen que pasar 30 años más para que la enseñanza asistida por ordenador recobre un interés especial, potenciado en gran parte por la aparición de los primeros ITSs. Innegablemente, hoy en día la preocupación por implantar sistemas de enseñanza virtual ha alcanzado uno de sus puntos más álgidos.

A continuación, en la tabla 1, se describen, muy brevemente, una serie de tutores inteligentes. Para cada uno, se indica la materia que trata y algunas de sus características más significativas.

Tabla 1: Sistemas tutores inteligentes

SCHOLAR [Carbonell, 70]	Materia: Geografía de Sudamérica
	Permite diferenciar y separar el conocimiento sobre "cómo enseñar" del conocimiento sobre la materia que se pretende estudiar.
	Usa una red semántica para representar el dominio de conocimiento.
	Plantea preguntas al alumno y permite que el alumno haga preguntas al sistema.
	Uno de sus puntos fuertes es el reconocimiento del lenguaje natural.
PLATO [Bitzer, 71]	Materia: Matemáticas
	Plantea una serie de conceptos que se quieren enseñar, e incorpora reconocedores de patrones para identificar si el estudiante ya los conoce o no.
	Sólo asesora al estudiante sobre los conceptos que no conoce, buscando el momento oportuno para mostrarle un ejemplo de aplicación.
MENO [Woolf, 84]	Materia: Lenguaje de programación Pascal
	Posee una base de datos con los errores comunes de los estudiantes que resuelven un problema.
	Plantea un problema al estudiante y reconoce los errores del programa solución al compararlo con las plantillas de la base de datos.
COACH [Selker, 94]	Materia: Lenguajes de programación
	Ambiente de ayuda interactiva.
	Modelo de usuario adaptativo, donde la asesoría que recibe el estudiante depende de su nivel de conocimiento.

3.2 Definición y arquitectura

Los tutores inteligentes surgen con la vocación clara de desarrollar procesos de enseñanza adaptados a las diferencias de los usuarios. De una forma amplia se puede definir un sistema tutor inteligente como un sistema de enseñanza asistida por ordenador, capaz de representar el conocimiento de un experto y dirigir la estrategia de enseñanza de sus usuarios. Pudiendo incluso, diagnosticar la situación en la que se encuentra el estudiante y de acuerdo a ello, ofrecer una solución que le permita progresar en su aprendizaje.

Los tutores inteligentes son capaces de observar el comportamiento de sus usuarios, reconocer los errores que éstos comenten e intervenir en caso de ser necesario. Todo ello con el fin último de enseñar una serie de conceptos.

A lo largo de los ya casi 30 años de investigación en este campo, se han obtenido avances más o menos estables como, por ejemplo, una arquitectura general aplicable a distintos dominios y tipos de sistemas, que gracias a la independencia de sus modelos permite ser integrada fácilmente con otros sistemas.

Los componentes generales de esta arquitectura son los que se muestran en la figura 2.

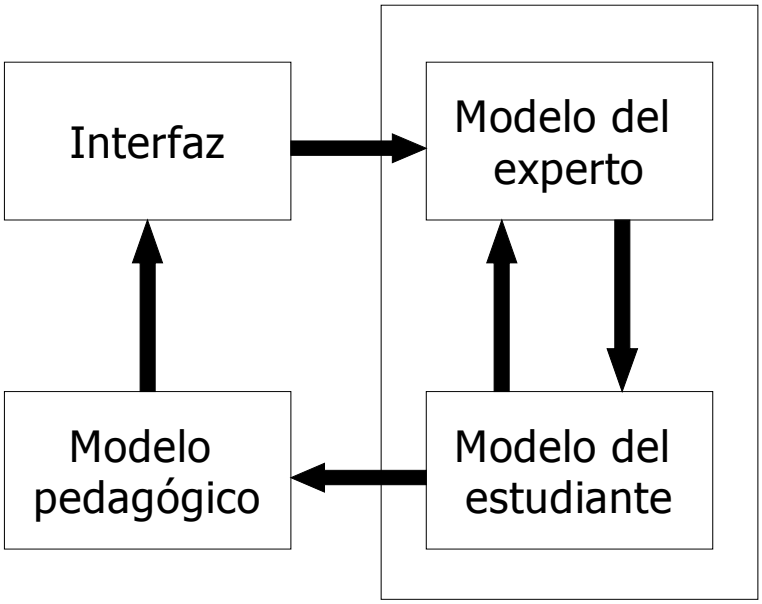


Figura 2: Arquitectura general para un ITS

La **interfaz** permite a los usuarios interactuar con el sistema. Presenta la información y obtiene las reacciones del alumno.

El **modelo del experto o del domino** representa la materia que se imparte. Este modelo se construye a partir de la información proporcionada por el experto y/o la que se extrae desde bases de conocimiento. Representa los conceptos que el alumno debe aprender, organizados de forma que la enseñanza de los mismos sea sencilla y eficaz.

El modelo **instructivo o pedagógico** define la estrategia que se va a seguir para transmitir el conocimiento a los usuarios. Este módulo es el encargado de adaptar el sistema al ritmo del alumno. La información que se muestra en cada momento dependerá de tres variables: decisión del currículo, selección del material didáctico y evaluación de la actividad del alumno.

El **modelo de estudiante** refleja cuánto conoce el alumno sobre el dominio de conocimiento, esto es, los conceptos que ha aprendido durante su interacción con el sistema. También, recoge las estrategias seguidas por éste durante su proceso de aprendizaje, lo cual permitirá obtener diagnósticos más completos y seleccionar el nivel de actuación adecuado.

El modelo de estudiante también guarda información sobre el material (ejercicios, ejemplos,...) que ha sido utilizado para enseñar al alumno. Esta información permitirá decidir de forma apropiada qué material se le ofrece más adelante. Obviamente, el modelo de estudiante es uno de los aspectos más esenciales dentro del diseño del sistema tutor.

Otros autores, como por ejemplo [Coltell, 99], organizan los anteriores componentes en dos macro-componentes: **componente hipermedia** y **componente tutor**. El primero, es simplemente, un tutorial multimedia tradicional. Mientras que el segundo mantiene toda la actividad inteligente del sistema.

Dentro del componente hipermedia se incluye el modulo de *interfaz*, el *hiperespacio* y un módulo de *control de navegación*. Este último se encarga de comunicar el componente hipermedia con el componente tutor, adaptando el hiperespacio a las ordenes procedentes del tutor. Estas órdenes dependen del seguimiento que el componente tutor realiza del alumno, durante el cual controla los conocimientos que éste adquiere en su interacción con el sistema. El esquema de esta arquitectura se muestra en la figura 3.

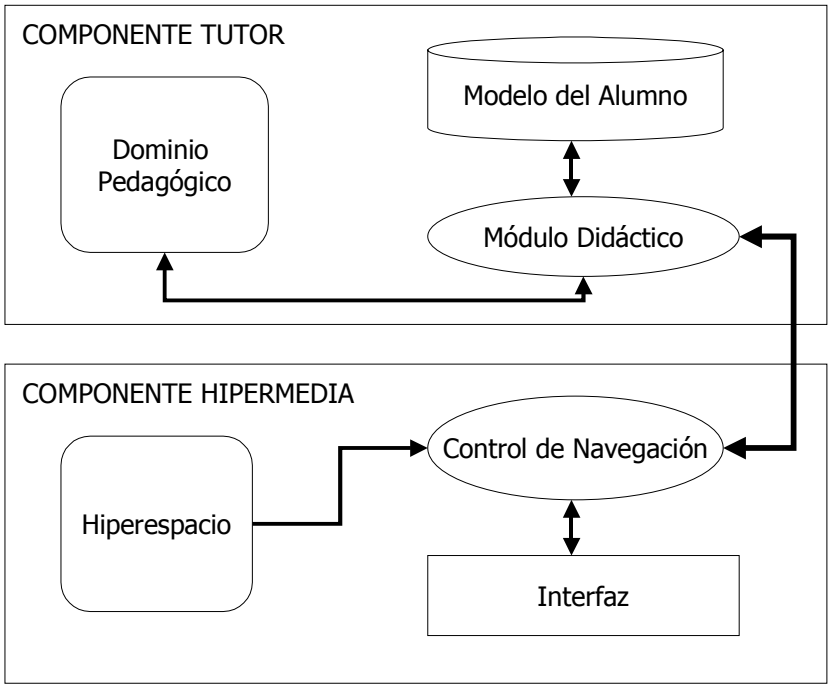


Figura 3: Arquitectura para un ITS [Coltell, 99]

De acuerdo a la estrategia de instrucción de los sistemas tutores inteligentes, se pueden distinguir dos tendencias:

- **Sistemas instructivos**, orientados a transmitir el conocimiento y guiar al estudiante durante el estudio del material ofrecido. Se trata de un aprendizaje fundamentalmente teórico y el sistema se comporta como un profesor transmisor.

- **Sistemas asesores**, orientados a apoyar al estudiante en la realización de una tarea que implica la aplicación de una serie de conocimientos teóricos. En este caso, el aprendizaje es práctico y el sistema se comporta como un profesor asesor que realiza

sugerencias al alumno para que mejore el desempeño de la tarea que en ese momento realiza.

3.3 Apoyo educacional

El objetivo principal de un sistema tutor inteligente es ayudar al estudiante en el aprendizaje de diversos conocimientos. Es, en cierto modo, un sistema semi-presencial de conocimiento, ya que el profesor está ausente y es el propio sistema el que guía al alumno en la asimilación de los diferentes conceptos. Para conseguir este objetivo, la mayoría de los sistemas tutores, aplican en mayor o menor grado, las siguientes tecnologías:

- **Seguimiento del Currículum:**

Esta técnica intenta suministrar al estudiante, con la mayor individualidad posible, un plan de aprendizaje que le ayude a encontrar su camino óptimo dentro de todo el material proporcionado por el sistema. El plan de aprendizaje se compone, fundamentalmente, de dos elementos:

a) Un conjunto de **secuencias de unidades de conocimiento** disponibles para aprender: lecciones, temas, ejemplos, etc.

b) Un **listado de tareas** para reforzar el aprendizaje. Entre estas tareas se encuentran: esquemas, ejercicios, problemas, exámenes, etc.

Existen dos tipos esenciales de secuencia: activa y pasiva. Una *secuencia activa* implica un objetivo de aprendizaje y suele proponer un conjunto de lecciones del temario para lograr el estado de conocimiento deseado.

Una *secuencia pasiva* empieza cuando el estudiante no es capaz de resolver un problema o responder a una cuestión correctamente, es pues, una técnica reactiva. El objetivo es entonces, ofrecer al estudiante cierto material que le pueda resultar útil para resolver el problema planteado, por ejemplo cubriendo algunas de sus necesidades de conocimiento.

El seguimiento o programación curricular puede realizarse a dos niveles de abstracción. A un nivel de abstracción alto, se realiza un seguimiento del conocimiento del alumno, que permite determinar el siguiente objetivo, concepto o lección a tratar. A un nivel de abstracción más bajo, el seguimiento curricular se realiza mediante tareas, de modo que el sistema establece la siguiente tarea de aprendizaje en función del objetivo actual.

Una dificultad importante reside en encontrar estrategias pedagógicas que de forma óptima establezcan cómo secuenciar la materia, cuándo resumir un concepto o explicar un ejemplo, cuándo proporcionar un ejercicio, cuándo presentar una analogía, etc.

Del mismo modo, no resulta fácil determinar cómo incorporar todo el conocimiento de un tutor humano, decidir cuántas estrategias son necesarias, cuándo aplicar una u otra, qué diferencias existen entre ellas, etc. Para solucionar todas estas dificultades derivadas de la elección y definición de estrategias pedagógicas, se hace necesario el uso de técnicas de Inteligencia Artificial [Murray, 99].

- **Análisis de las Soluciones:**

Es necesario pronosticar la evolución del alumno y su correspondiente solución durante la realización de una tarea. El análisis debe decidir si la solución dada por el alumno a un problema o ejercicio es correcta. En caso negativo, se debe encontrar qué es exactamente lo que está mal o incompleto. A partir de ahí, el sistema debe identificar qué conocimiento le falta al alumno para que haya cometido ese error o qué conocimiento erróneo le ha llevado a esa solución desatinada.

El análisis realizado es un análisis inteligente, que no se detiene en el error, sino que busca y profundiza en sus causas, tal y como lo haría un experto. Algunos autores [Vernet, 01] incorporan sistemas de predicción capaces de aventurar la evolución del alumno, por ejemplo, pronosticando si éste aprobará o no la asignatura, e incluso anticipando qué nota sacará al final del curso.

- **Apoyo a la Solución:**

Dependiendo del nivel de conocimiento del alumno y de la dificultad de la tarea que realiza, puede resultar muy conveniente asistir o asesorar al alumno de forma interactiva. La idea es ofrecer al estudiante ayuda inteligente en cada paso de la resolución del problema. El nivel de ayuda puede variar, desde señalar qué está mal en cada paso, hasta ejecutar el siguiente paso por el estudiante. Otro tipo de apoyo a la solución consiste en proporcionar casos de solución satisfactorios para problemas muy similares al actual.

CAPÍTULO 5

Web Semántica

Resumen

La Web Semántica es un proyecto ambicioso que pretende incrementar las posibilidades de la Web añadiendo semántica en la representación y tratamiento de sus documentos. En el presente capítulo se exponen los objetivos de la Web Semántica, sus principales características y los beneficios que aportaría. También se enumeran y describen algunas de las tecnologías que deberían usarse conjuntamente para hacer de la Web Semántica una realidad.

Tabla de contenidos

Web Semántica

1. INTRODUCCIÓN

Hoy en día, el hipermedia adaptativo constituye una parte importante de la Web. Los métodos de personalización y adaptación al usuario han llegado a ser cruciales para manejar de forma eficiente la información en Internet. Al mismo tiempo, las tecnologías y estándares desarrollados en el campo de la Web Semántica ofrecen soluciones flexibles que pueden aplicarse también a las necesidades de los sistemas hipermedia adaptativos.

En [Aroyo, 04] se expone una visión de cómo el desarrollo de la Ingeniería del Conocimiento en el contexto de la Web Semántica puede contribuir a mejorar la aplicabilidad y reusabilidad de los sistemas adaptativos, aumentando, también, su capacidad para compartir.

De acuerdo a este enfoque, proponen una arquitectura de hipermedia adaptativo basada en web, abierta y modular. Resaltan la necesidad de una separación estricta entre el modelo de dominio, el modelo de adaptación y el modelo de usuario, la conveniencia de proporcionar ricas descripciones semánticas basadas en ontologías, y la de mantener modelos genéricos, incluso estándares, que faciliten la comunicación entre distintos sistemas hipermedia adaptativos (especialmente para el modelo de usuario).

En definitiva, la Web Semántica permite aumentar la adaptación y la interoperabilidad entre los sistemas hipermedia adaptativos, a la vez que se realimenta de las capacidades de dichos sistemas. Según [Aroyo, 04] éste constituye el primer paso hacia la definición de una nueva clase de entorno hipermedia adaptativo basado en web semántica.

2. PROBLEMÁTICA

La Web actual es un espacio preparado para el intercambio de información creado para el consumo humano. Esto es, las páginas son creadas por personas para ser entendidas por personas, y por lo tanto, se diseñan considerando únicamente a los usuarios potenciales que van a visitarlas.

Es indudable que las ventajas que ofrece Internet son enormes a la hora de buscar información. No obstante, adolece de una manera precisa de encontrar información, puesto que la mayoría de los navegadores realizan la búsqueda mediante palabras clave que aparecen en el código HTML de las páginas. Concretamente, la mayoría de los agentes o buscadores web, aplican un algoritmo de emparejamiento de patrones, que tiene en cuenta el número de veces que se hace referencia en cada página candidata a formar parte del resultado de la búsqueda.

Normalmente gastamos mucho tiempo en seleccionar la información que puede sernos útil, navegando por páginas no deseadas, hasta encontrar información relevante. Esto se debe a que la única información que recuperamos con el sistema de búsqueda actual son conceptos descontextualizados.

Por ejemplo, si en el buscador escribimos la palabra "mesa", podemos encontrar una página de cocina, otra con información sobre juegos de mesa, la página personal de individuo apellidado Mesa, restaurantes, tenis de mesa, etc. Cuando la finalidad de quién consulta podía haber sido, perfectamente, obtener imágenes y precios de mesas para amueblar su salón.

Otra carencia es que no se puede diferenciar el tipo de información que contiene la página: información personal, académica, comercial, etc. Del mismo modo que no todas las páginas proporcionan igual cantidad de información, ni al mismo nivel. Esto, unido al hecho de que los agentes web no se diseñan para "comprender" la información que inspeccionan, hace que las consultas de información en Internet no sean tan productivas como debieran.

3. EL SALTO EVOLUTIVO DE LA WEB

La Internet que había imaginado en 1989 el creador de la *World Wide Web*, Tim Berners-Lee, no era exactamente lo que hoy conocemos. De hecho, ya entonces Berners-Lee pensaba en algo más revolucionario, algo muy parecido a los que hoy denominamos Web Semántica. Cada vez son más los autores que ven en la semántica el futuro de la Web. Una web que facilitará la localización de recursos, la comunicación entre sistemas y programas, que nos ayudará a gestionar nuestro día a día, hasta llegar a niveles inimaginables.

La idea general de la Web Semántica es transformar la red en un cerebro global. Los precursores de esta idea son Francis Heylighen, Johan Bollen y Cliff Joslyn [Heylighen, 96] [Heylighen, 99] [Bollen, 96][Joslyn, 96].

La Web Semántica persigue llegar a un punto en el que los servidores web tengan la capacidad de "entender" el contenido de los documentos que almacenan. A la vez, que puedan estructurar y gestionar la información a partir de una valoración semántica de dicho contenido. Dicho de otro modo, la Web Semántica va a permitir a los ordenadores interpretar la información que aparece en Internet sin necesidad de intervención humana. Además, todo ello sin requerir la aplicación de Inteligencia Artificial, ya que la semántica está en las propias páginas.

Esto se logra enlazando y definiendo los datos de forma que puedan ser automatizados, integrados y reutilizados entre aplicaciones sin que exista un individuo que inicie los procesos. Esto significa que las máquinas (ordenadores personales o cualquier otro dispositivo conectado a Internet) podrán realizar, apenas sin intervención humana, infinidad de tareas destinadas a simplificar nuestra vida.

Es necesario resaltar que la Web Semántica no es una web aparte sino una extensión de la Web actual, en la que la información tiene un significado bien definido, posibilitando que los ordenadores y las personas trabajen en cooperación [Berners-Lee, 01].

Hasta ahora, la Web estaba concebida más para proporcionar documentos que para manipular datos y/o procesar información de manera automática. Se han dado ya los primeros pasos para incluir la Web Semántica en la estructura de la Web existente. Sin embargo, aún queda por delante un largo camino, donde el primer paso es tomar conciencia de que nos encontramos ante una potente herramienta facilitadora de la

comunicación, que además mantiene los principios que han hecho un éxito de la Web actual: descentralización, compartición, compatibilidad, apertura al crecimiento y usos no previstos de antemano.

En un futuro, no lejano, estos desarrollos introducirán prestaciones nuevas e importantes, al lograr que las máquinas multipliquen su capacidad de procesar y comprender los datos que hoy tan sólo se exhiben en pantalla. En cierta forma se habla de un reordenamiento, de ordenar el caos.

4. BENEFICIOS DE LA WEB SEMÁNTICA

Obviamente los agentes web mejorarán su eficacia cuando los contenidos de la Web tengan una semántica explícita. Al mismo tiempo, las búsquedas muy precisas devolverán mejores resultados cuando la Web Semántica sea una realidad universal, ya que será posible combinar informaciones que residen en diferentes páginas, entre las que ahora no existe ninguna conexión.

Los agentes web, no sólo encontrarán la información de forma precisa, sino que, además, podrán realizar inferencias automáticas para buscar información relacionada con la página actual de acuerdo a los requerimientos de la consulta del usuario.

Además, se potenciará el desarrollo de aplicaciones de comercio electrónico, ya que las anotaciones de información seguirán un esquema común, también compartido por los buscadores web. Por ejemplo, las empresas podrán intercambiar sus datos siguiendo estos esquemas consensuados.

Al mismo tiempo, si está correctamente diseñada, la Web Semántica puede ayudar a la evolución del conocimiento humano en general. Por ejemplo, facilitando la colaboración entre equipos investigadores de un mismo tema que se encuentran en lugares muy distantes del planeta. Entre los campos de aplicación donde la Web Semántica puede tener utilidad, es posible resaltar los siguientes [Castell, 02]:

- Búsqueda de información
- Comercio electrónico
- Gestión del conocimiento corporativo
- Procesamiento del lenguaje natural
- Enseñanza virtual
- Librerías digitales
- Turismo
- Patrimonio cultural

5. TECNOLOGÍA DE LA WEB SEMÁNTICA

Para llegar a esta nueva web hay que dar una serie de pasos que no son sencillos. Uno de los retos principales que afronta la Web Semántica es el de proveer un lenguaje que permita exportar a la Web las reglas de cualquier sistema de representación del conocimiento. Esto es, con otras palabras, añadir lógica a la Web.

La idea recurrente es que los datos puedan ser utilizados y "comprendidos" por los ordenadores sin supervisión humana. Así pues, se trata de convertir la información en conocimiento, referenciando datos dentro de las páginas web a metadatos (datos sobre datos) con un esquema consensuado.

Si bien los actuales documentos en HTML (*Hypertext Markup Language*) poseen cierta cantidad de metadatos que permite su indexación en buscadores, no es suficiente porque se refieren al documento en general y no a cada uno de sus componentes. Es necesario un lenguaje entendible por las computadoras que permita asignar más información sobre sí mismo al documento.

El desarrollo de la Web Semántica pasa por la adopción de diferentes tecnologías. En la figura 1 se muestra una estructura en capas para la Web Semántica utilizada por Tim Beernes-Lee en la conferencia XML-2000 (Washington DC, Diciembre, 2000) [Beernes-Lee, 00].

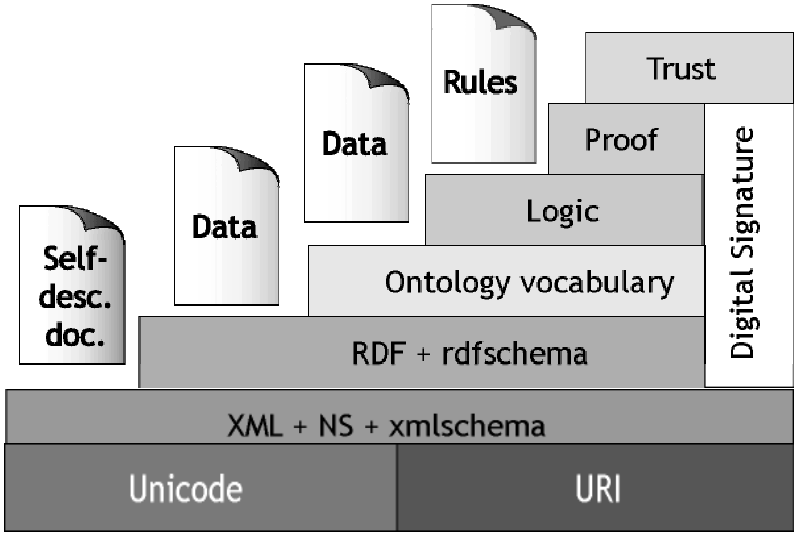


Figura 1: Estructura en capas para la Web Semántica [Beernes-Lee, 00]

Las capas se superponen unas sobre otras, partiendo como base de un sistema de codificación universal (Unicode) e identificadores para los recursos de la Web (URIs).

Unicode [Unicode] es un sistema de codificación de ocho bits, que ofrece un único código para cada carácter sin importar la plataforma, el programa ni el lenguaje. La universalización del juego de caracteres Unicode evita, a todos los niveles, los problemas que tienen lugar cuando diversos juegos de caracteres chocan: contraseñas, direcciones y dominios con eñe o acentos, sin hablar de las incompatibilidades con las que se encuentran chinos, árabes y la gran mayoría de la humanidad que utiliza grafías no latinas.

Sobre la primera capa se sitúa el lenguaje XML que permite añadir estructura al documento. Y encima, el marco RDF capaz de asociar significado a la estructura, de una forma procesable por las máquinas. Un vocabulario ontológico constituye la siguiente capa y junto con la anterior permite que el documento se conozca a sí mismo, a través de los metadatos introducidos.

Encima una capa lógica permite la inclusión de reglas para tratar convenientemente el conocimiento representado, y junto a las dos capas que le siguen pretende lograr la

confianza en la Web (estas tres últimas capas aún están en discusión). Obviamente, mecanismos de seguridad son necesarios durante casi todo el proceso, destacando la firma digital para evitar problemas de identidad.

5.1 XML (*eXtensive Markup Language*)

Está basado en el anterior estándar SGML (*Standard Generalized Markup Language*), que data de 1986, pero que empezó a gestarse a principios de los años 70, y a su vez basado en el GML (*Geographic Markup Language*) creado por IBM [IBM] en 1969.

Esto significa que aunque XML [Harold, 98] [Goldfarb, 00] pueda parecer moderno, sus conceptos están más que asentados y aceptados desde hace bastante tiempo. Motivo por el cual, con pocos años de vida, en febrero de 1998 se convirtió en un estándar del World Wide Web Consortion [W3C].

SGML [Goldfarb, 90] proporciona un modo consistente y preciso de aplicar etiquetas para describir las partes que componen un documento, permitiendo además el intercambio de documentos entre diferentes plataformas. De todas sus características pueden extraerse tres básicas:

1. **Formal**. Permite establecer la validez de los documentos.
2. **Estructurado**. Es capaz de manejar documentos complejos.
3. **Ampliable**. Facilita la gestión de grandes depósitos de información.

Sin embargo, el problema que se atribuye a SGML es su excesiva dificultad. Así que, manteniendo su misma filosofía, de él derivó XML [XML] como un subconjunto simplificado. Como su padre, XML es un metalenguaje, esto es, un lenguaje de marcas con el que se pueden crear lenguajes de marcas para documentos concretos. Los elementos que lo componen pueden dar información sobre lo que contienen, sin limitarse necesariamente a su estructura física o presentación, como ocurre en HTML [Gulbransen, 98].

En XML las etiquetas sólo sirven para delimitar los elementos de datos, dejando la interpretación de los mismos a la aplicación que los lee. En consecuencia, en XML se puede especificar cualquier dominio, convirtiéndose en un excelente mecanismo de intercambio de datos entre aplicaciones.

Un documento XML está bien formado si cumple las especificaciones sintácticas del lenguaje, mientras que para ser válido, debe seguir, además, una estructura y una semántica determinada por un esquema de XML.

Un esquema define la estructura, contenido y semántica de un conjunto de documentos XML. Determina qué elementos pueden contener y en qué orden, cuál puede ser el contenido de los elementos y qué atributos se pueden definir de los elementos. Un documento es válido respecto a un esquema si cumple todas las reglas definidas en él. La validación es el proceso de comparación.

Un DTD (*Document Type Definition*) es una definición de los elementos que puede haber en el documento XML, la relación que puede existir entre ellos, sus atributos, los

posibles valores de éstos, etc. En definitiva es una especie de definición de la gramática del documento. Así, cuando se procesa cualquier información formateada mediante XML, lo primero que se debe hacer es comprobar si está bien formada, y luego, en el caso de que incluya o haga referencia a un DTD, comprobar que sigue sus reglas gramaticales.

Los DTDs están dejando de usarse porque presentan algunos inconvenientes: tienen su propio lenguaje (no se pueden usar las herramientas de XML), todas las declaraciones son globales y presentan limitaciones para definir los tipos asociados a elementos y atributos. En su lugar, se impone el uso de **XML Schema**, una aplicación XML para definir otras aplicaciones XML. XML Schema fue aprobado como una recomendación W3C en Mayo de 2001 [W3C].

Un XML Schema es algo similar a un DTD, es decir, define qué elementos puede contener un documento XML, cómo están organizados, y que atributos y de qué tipo pueden tener sus elementos. Sin embargo, permite declarar variables locales y proporciona una estructura de tipos mucho más rica. XML Schema admite tipos simples y complejos. Los tipos simples permiten definir patrones y especificar valores por defecto, conjuntos de valores posibles y rangos de valores aceptables. Los tipos complejos permiten definir conjuntos de opciones alternativas, atributos requeridos, grupos de elementos o atributos con nombre (para reutilizarlos), elementos basados en otros, etc. Hay cuatro tipos básicos según contengan: sólo otros elementos, elementos y texto, sólo texto o estén vacíos.

Para poder combinar adecuadamente documentos de varios esquemas y facilitar la reutilización y gestión en esquemas grandes, en XML Schema se incorpora el concepto de espacio de nombres (***namespace,* NS**). Un espacio de nombres es una colección de elementos relacionados, identificados por el nombre del espacio. Cada esquema define la estructura de documentos XML y crea un espacio de nombres al que se añaden por defecto todos los elementos globales (también se pueden añadir los locales, e incluso los atributos). El espacio de nombres afecta al elemento donde se declara y a todos los elementos contenidos en él.

5.2 RDF (*Resource Description Framework*)

RDF [RDF] proporciona una estructura adecuada para la descripción de recursos, entendiendo por recurso cualquier cosa en torno a la red: páginas, personas, dispositivos, etc. RDF ofrece una estructura semántica que no es ambigua y que permite la codificación, el intercambio y el procesamiento automático de metadatos normalizados.

Concretamente RDF usa la sintaxis del lenguaje XML para el intercambio y procesamiento de los metadatos. No obstante, es conveniente indicar que XML es sólo una sintaxis posible para RDF y que pueden surgir formas alternativas para representar el mismo modelo de datos (la especificación del modelo es independiente de su sintaxis).

El objetivo general de RDF es definir un mecanismo de descripción de recursos que no cree ninguna asunción sobre un dominio de aplicación particular, ni defina a priori la

semántica de algún dominio de aplicación. Sin embargo el mecanismo debe ser adecuado para describir información sobre cualquier dominio.

El modelo de datos RDF está basado en sentencias o tríos. Cada sentencia está compuesta por un recurso, una propiedad y un valor. Las sentencias se establecen sobre objetos o recursos y a los recursos se le asocian propiedades con valores.

Todas las cosas descritas por expresiones RDF se denominan **recursos**. Un recurso puede ser una página web completa, una parte de una página, una colección completa de páginas, un libro impreso, etc. Los recursos se designan siempre por URIs (*Uniform Resource Identifiers*). Cualquier cosa, física o abstracta, puede tener un URI, incluso aunque no sea accesible desde la red. Las URL (*Uniform Resource Locator*) son un tipo particular de URI.

Una **propiedad** es un aspecto específico, característica, atributo, o relación utilizado para describir un recurso. Cada propiedad tiene un significado específico, define sus valores permitidos, los tipos de recursos que puede describir, y sus relaciones con otras propiedades.

Un recurso específico junto con una propiedad, más el valor de dicha propiedad para el recurso es una **sentencia RDF**. Se trata pues de frases del tipo: *sujeto* tiene *predicado* de *objeto*. Con la intención de poder validar si el predicado (propiedades) es adecuado para el sujeto (recurso) y si el objeto (valor) es adecuado para el predicado (propiedad) surge **RDF Schema**. Éste proporciona un mínimo de funcionalidades para restringir los metadatos. No obstante, en algunas aplicaciones esto puede no ser suficiente, por lo que se necesitan lenguajes para expresar ontologías, entre los que destacamos **OWL**.

5.3 OWL (*Web Ontology Language*)

OWL [Smith, 04] es un lenguaje para definir e instanciar ontologías web. Es una recomendación de W3C desde Febrero de 2004 [WC3] y tiene como punto de partida las experiencias previas realizadas con DAML+OIL en las cuales se inspiraron sus creadores. El objetivo de OIL (*Ontolgy Inference Layer*) es definir ontologías, mientras que DAML (*DARPA Agent Markup Language*) intenta definir un lenguaje para definir otros lenguajes que permitan comunicar agentes.

Una ontología OWL puede incluir definiciones de clases, propiedades y sus instancias. Dada una ontología OWL, la semántica formal de OWL especifica cómo derivar sus consecuencias lógicas con independencia del dominio representado en la ontología.

OWL se divide en tres sublenguajes OWL-Lite, OWL-DL y OWL-Full, cada uno de los cuales proporciona un conjunto definido sobre el que trabajar, siendo el más sencillo OWL-Lite y el más completo OWL-Full. OWL-Lite permite definir jerarquías de clasificación y restricciones simples. OWL-DL ofrece la máxima expresividad manteniendo completitud computacional y decidibilidad. Y OWL-Full ofrece la máxima expresividad manteniendo la libertad sintáctica de RDF, pero sin garantías computacionales.

Los constructores de OWL-Lite son los que se muestran a continuación. A éstos se añaden algunos más, por ejemplo para trabajar con axiomas, en OWL-DL y OWL-Full.

RDF Schema Features:

- *Class (Thing, Nothing)*
- *rdfs:subClassOf*
- *rdf:Property*
- *rdfs:subPropertyOf*
- *rdfs:domain*
- *rdfs:range*
- *Individual*

(In)Equality:

- *equivalentClass*
- *equivalentProperty*
- *sameAs*
- *differentFrom*
- *AllDifferent*
- *distinctMembers*

Datatypes

- *xsd datatypes*

Class Intersection:

- *intersectionOf*

Property Characteristics:

- *ObjectProperty*
- *DatatypeProperty*
- *inverseOf*
- *TransitiveProperty*
- *SymmetricProperty*
- *FunctionalProperty*
- *InverseFunctionalProperty*

Poperty Restrictions:

- *Restriction*
- *onProperty*
- *allValuesFrom*
- *someValuesFrom*

Restricted Cardinality:

- *minCardinality* (0 or 1)
- *maxCardinality* (0 or 1)
- *cardinality* (0 or 1)

Header Information:

- *Ontology*
- *imports*

Versioning:

- *versionInfo*
- *priorVersion*
- *backwardCompatibleWith*
- *incompatibleWith*
- *DeprecatedClass*
- *DeprecatedProperty*

Annotation Properties:

- *rdfs:label*
- *rdfs:comment*
- *rdfs:seeAlso*
- *rdfs:isDefinedBy*
- *AnnotationProperty*
- *OntologyProperty*

5.4 Otros elementos

5.4.1 Encriptación y firma digital

La firma digital es un procedimiento técnico que, basándose en técnicas criptográficas, trata de dar respuesta a cuatro necesidades esenciales:

– Identidad de la persona implicada en el proceso de comunicación.

– Integridad de la información enviada durante el proceso de transmisión.

– No rechazo en origen, esto es, que el emisor del mensaje no pueda negar en ningún caso que el mensaje ha sido enviado por él.

– Confidencialidad, que no es un requisito esencial de la firma digital sino accesorio de la misma. La confidencialidad implica que el mensaje no haya podido ser leído por terceras personas distintas del emisor y del receptor durante el proceso de transmisión del mismo.

Obviamente la Web Semántica necesita de un sistema generalizado de firma y encriptación que permita definir la autoría de contenidos, fechas de modificación, quién puede acceder a ellos, quién puede modificarlos, quién puede modificar las reglas de acceso, etc.

En concreto, **XML Signature** permite validar las características de *integrity* y *nonrepudiation* de los mensajes, y firmar digitalmente cualquier parte de un documento XML. Mientras que **XML Encryption** da soporte a la característica de *confidenciality*, permitiendo ocultar el contenido sensible, de forma que sólo el destinatario pueda leer esa parte del documento XML.

5.4.2 Agentes

Un agente es una entidad software que funciona continua y autónomamente en un medio particular a menudo habitado por otros agentes y procesos, sin requerir de guía

constante o intervención humana. En otras palabras, un agente es un asistente personal que está dentro de la computadora y que cumple varios roles en representación de una función específica para ello.

La Web Semántica se basa en la estandarización de todos sus datos. Todo en la Web debería presentarse en un mismo formato comprensible por una nueva generación de agentes inteligentes que clasificarán la información de una manera más eficiente para devolver resultados más precisos ante un servicio de búsqueda. La semántica facultará a los agentes para describir unos a otros la función exacta que realizan, y qué datos han de recibir para ello.

5.5 Ontologías en la Web Semántica

Junto con las tecnologías descritas en la sección anterior y especialmente si hablamos de intercambio de conocimiento no podemos dejar de considerar las ontologías como un soporte fundamental para la Web Semántica [Lozano, 01].

El término ontología en informática toma su nombre de una analogía con el término filosófico del mismo nombre y hace referencia al intento de formular un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, con el objetivo de facilitar la comunicación y compartir de la información entre diferentes sistemas.

En [Maedche, 02] se define una ontología como una teoría lógica formada por un vocabulario y un lenguaje lógico, donde se formalizan signos de un dominio de interés que describen cosas en el mundo real, permitiendo una correspondencia entre los signos y las cosas tan exacta como sea posible. No obstante, existen otras muchas definiciones de ontología, entre las que podemos extraer las que se muestran en la tabla 1.

Tabla 1: Noción de Ontología

Autor/Año	Definición de Ontología
[Bylander, 88]	"...la tarea de representar el conocimiento con el propósito de resolver un problema se ve fuertemente afectada por la naturaleza del problema y la estrategia de inferencia a aplicar..."
[Alberts, 93]	"La ontología describe una taxonomía de conceptos para una tarea o dominio que define la interpretación semántica del conocimiento"
[Gruber, 95]	"Una ontología es una especificación explícita de una conceptualización"
[Van Heijst, 97]	"Una ontología es una especificación explícita de una conceptualización a nivel de conocimiento, (...) a la cuál le puede afectar el dominio y tareas específicos para los que está pensada"
[Stuber, 98]	"Una ontología es una especificación explícita y formal sobre una conceptualización compartida"

Existen distintos tipos de ontologías [Maedche, 02]:

- *Top-level ontologies*: Describen conceptos generales independientes de un dominio particular.
- *Domain ontologies y Task ontologies*: Describen respectivamente el vocabulario relacionado con un dominio y una tarea genérica mediante la especialización de una ontología *top-level*.

– *Application ontologies*: *S*on más específicas, ya que trabajan con dominios y actividades concretas. Especializan *domain ontologies* y *task ontologies*.

En la figura 2 se muestran el espectro de las ontologías, diferenciando las propiedades necesarias, típicas y deseables para éstas:

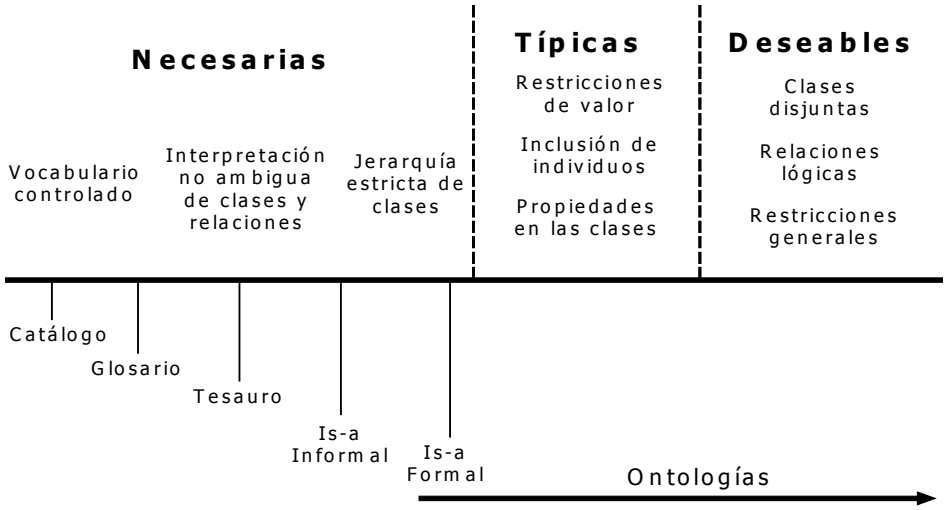


Figura 2: El espectro de las ontologías [McGuiness, 01]

Las ontologías tienen una serie de componentes para representar el conocimiento del dominio, de forma consensuada, reutilizable y legible por los ordenadores.

- Conceptos: Son las ideas básicas que se pretenden formalizar.
- Instancias: Se utiliza para representar objetos de un determinado concepto.
- Funciones: Tipo concreto de relación que identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- Relaciones: Representan la interacción y enlace entre los conceptos del dominio.
- Axiomas: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Junto con la herencia permiten inferir conocimiento no explícito.

De forma resumida podemos decir que las ontologías son colecciones de enunciados redactados en un lenguaje, por ejemplo RDF, que define las relaciones entre conceptos y especifica reglas lógicas para razonar con ellos. Los ordenadores "comprenderán" el significado de los datos semánticos de una página siguiendo vínculos con ontologías especificadas.

Entre las ventajas o utilidades de establecer una ontología podemos citar las siguientes:

– Compartir conocimiento común sobre la estructura de las cosas (ontologías públicas y extensibles).

– Re-usar el conocimiento del dominio (evolución de ontologías).

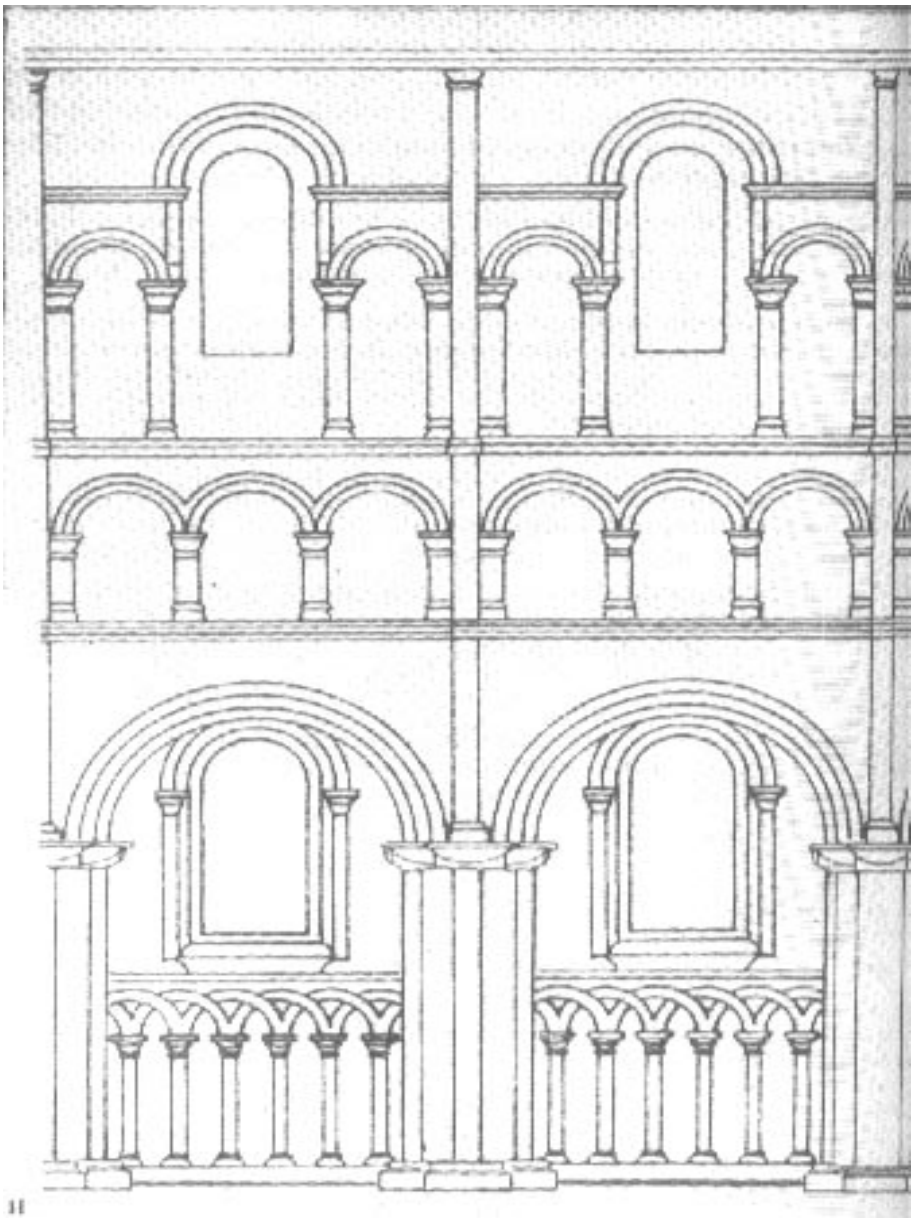
– Explicitar suposiciones sobre el dominio.

– Separar el conocimiento del dominio del conocimiento operacional.

– Posibilitar el análisis del conocimiento del dominio.

Módulo II

El modelo SEM-HP

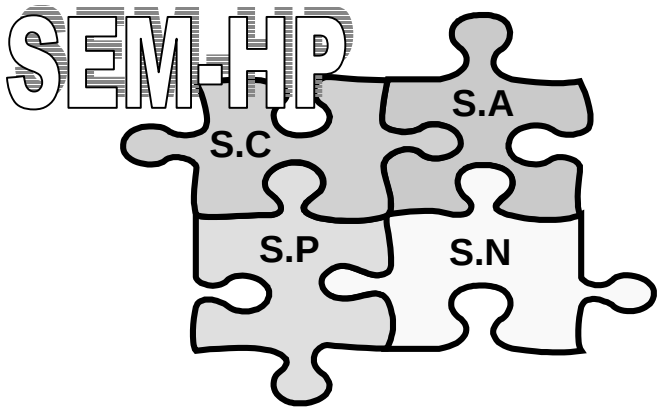


Los Pilares de la Tierra

Módulo II

CAPÍTULO 6

El Modelo SEM-HP



Resumen

SEM-HP es un modelo para el desarrollo de sistemas hipermedia que respalda la naturaleza evolutiva de su construcción, mantenimiento y navegación. Para ello, el sistema hipermedia es modelado mediante un conjunto de sistemas interrelacionados y en interacción. En el presente capítulo se describe el proceso de desarrollo propuesto en el modelo y los cuatro sistemas que componen su arquitectura: los tres primeros, memorización, presentación y navegación, se encuentran desarrollados en [García, 01], mientras que el cuarto, el sistema aprendizaje responsable de la adaptación al usuario, constituye el eje principal de esta tesis.

Tabla de contenidos

El Modelo SEM-HP

1. INTRODUCCIÓN

El modelo SEM-HP desarrollado en la tesis doctoral de García-Cabrera [García, 01c] supone una innovadora alternativa para la construcción de sistemas hipermedia respecto a los modelos tradicionales, que en su mayoría no garantizan una correspondencia entre estructura y funcionalidad, ni asisten al autor en el proceso de desarrollo y mantenimiento posterior.

SEM-HP es un modelo **SEM**ántico, **S**istémico y **E**volutivo que permite el desarrollo de sistemas **H**i**P**ermedia. En él, el proceso de diseño y construcción del sistema hipermedia está basado en un modelo cognitivo [García, 97][García, 00], y el autor puede caracterizar el dominio de conocimiento mediante sus propias ontologías. Además el modelo facilita y hace flexible la construcción, mantenimiento y navegación de los sistemas hipermedia, que son capaces de integrar los continuos cambios y adaptaciones, al incorporar mecanismos evolutivos.

El modelo SEM-HP (figura 1) proporciona al autor tres elementos para la creación de sistemas hipermedia evolutivos y adaptativos: un proceso de desarrollo, una arquitectura y una herramienta de autor.

Figura 1: Modelo SEM-HP

El **proceso de desarrollo** establece las pautas a seguir para la creación del sistema desde un enfoque de ingeniería del software. La **arquitectura** describe los modelos de representación utilizados para capturar cada una de las fases del proceso de desarrollo. Y la **herramienta de autor** facilita la creación del sistema de acuerdo a la arquitectura y el proceso de desarrollo propuestos en el modelo.

2. EL PROCESO DE DESARROLLO

A pesar de sus características especiales, el diseño e implementación de un sistema hipermedia no deja de ser un proceso de desarrollo del software y como tal, la calidad del producto obtenido va a depender de la calidad del proceso efectuado para obtenerlo

[Medina, 02]. Por lo tanto, creemos que es conveniente aplicar un proceso de ingeniería del software en el desarrollo de estos sistemas.

El proceso de desarrollo propuesto en el modelo SEM-HP (figura 2) se divide en cuatro fases inherentes al diseño de un sistema conceptual y navegacional: memorización, presentación, navegación y aprendizaje. Cada una de estas fases genera como resultado un sistema de la arquitectura.

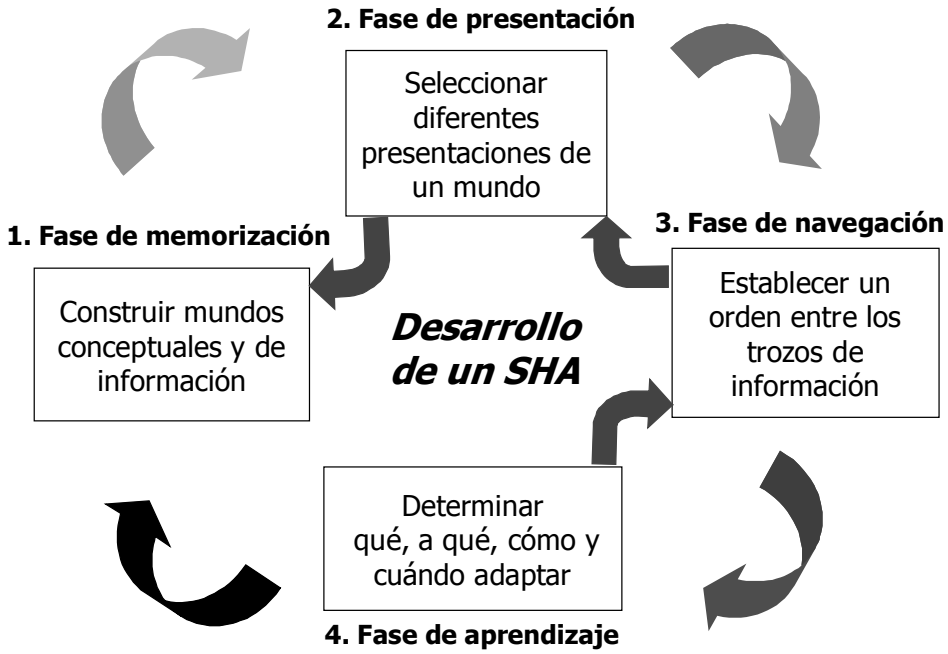


Figura 2: Proceso de desarrollo en SEM-HP

En la **fase de memorización** el autor define el dominio de información que recorrerán los usuarios de su sistema. Con objeto de hacer explícitas las relaciones semánticas entre los ítems de información, el autor también especifica el dominio conceptual subyacente.

De acuerdo a [García, 01c] el dominio de información y el dominio conceptual de un sistema hipermedia quedan perfectamente determinados a través de las siguientes definiciones:

Def 6.1 [Concepto] Un concepto es una idea, pensamiento o abstracción que puede ser etiquetado por el autor con el fin de hacer explícito su conocimiento y hacerlo comprensible.

Def 6.2 [Ítem] Un ítem es cualquier trozo de información identificable en un sistema hipermedia.

Def 6.3 [Dominio conceptual] Un dominio conceptual es un conjunto de conceptos con los que se pueden identificar los distintos ítems de información ofrecidos en un sistema hipermedia, y el conjunto de asociaciones semánticas que se pueden establecer entre ellos.

Def 6.4 [Dominio de información] Un dominio de información es el conjunto de trozos de información identificados con conceptos pertenecientes a un dominio conceptual concreto, incluyendo la asociaciones que los identifican.

En la **fase de presentación** el autor selecciona diferentes presentaciones o vistas del dominio de conocimiento forjado en la fase anterior.

En la **fase de navegación** el autor establece cómo el usuario puede navegar la información ofrecida.

Y, finalmente, en la **fase de aprendizaje** el autor resuelve los aspectos relacionados con la adaptación, respondiendo a las preguntas: ¿A qué?, ¿qué?, ¿cómo? y ¿cuándo adaptar?. Como resultado, establece los mecanismos necesarios para que posteriormente, durante su funcionamiento, el propio sistema sea capaz de ajustarse a las características e intereses de cada usuario.

Estas *fases* no son secuenciales sino *iterativas*, es decir, el autor puede regresar a una fase anterior siempre que lo necesite. Además, este proceso implica un desarrollo *evolutivo*, ya que sus cuatro fases son capaces de integrar, de una forma fácil, flexible y consistente, los cambios que el autor realiza en la estructura del sistema correspondiente.

3. ARQUITECTURA DE SEM-HP

La arquitectura propuesta por el modelo SEM-HP se estructura en capas realizando una doble división [García, 02] tal y como se muestra en la figura 3.

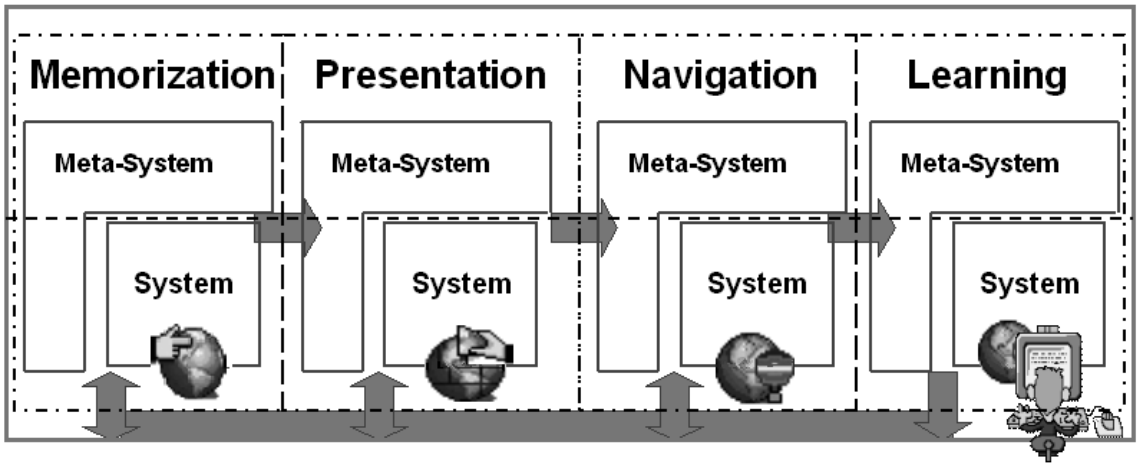


Figura 3: Arquitectura SEM-HP

La **división vertical** contempla cuatro sistemas interrelacionados y en interacción a los que se denomina **Sistemas de Memorización** (S.M), **Presentación** (S.P), **Navegación** (S.N) **y Aprendizaje** (S.A). Cada uno de estos sistemas registra algunas de las características del modelo, y por tanto, ofrece cierta funcionalidad al resto de sistemas y, en última instancia, a los usuarios del mismo.

La **división horizontal** [García, 01][García, 01b] distingue dos capas dentro de cada una de las anteriores: sistema y meta-sistema. Cada capa representa un nivel de abstracción diferente. El menos abstracto (**sistema**) contiene los modelos de

representación definidos por el autor durante la correspondiente fase de desarrollo, mientras que el más abstracto (**meta-sistema**) incluye los mecanismos evolutivos que permitirán integrar y propagar los cambios realizados por el autor en los elementos de dicho sistema.

Los mecanismos evolutivos incluidos dentro del meta-sistema son fundamentalmente tres: acciones evolutivas, restricciones y propagación del cambio. El autor interacciona con el meta-sistema para construir y modificar el sistema hipermedia. Concretamente, para realizar un cambio en el sistema, el autor selecciona y ejecuta la **acción evolutiva** correspondiente. Con el objeto de garantizar la integridad del sistema a lo largo de toda su vida, el meta-sistema deniega la realización de los cambios inconsistentes. Así, una acción evolutiva solo se ejecuta si satisface una serie de **restricciones** impuestas por el propio modelo (*restricciones del sistema*) y por el autor (*restricciones de autor*).

Finalmente, al modificar un elemento de uno de los cuatros sistemas de la arquitectura, puede surgir la necesidad de modificar otros elementos de ese mismo sistema (*propagación interna*) o incluso de algún otro (*propagación externa*). En ambos casos, la **propagación del cambio** es realizada automáticamente en el modelo, garantizando con ello, una co-evolución coherente de la arquitectura completa del sistema hipermedia [Medina, 03d].

> **Def 6.5 [Acción evolutiva]** Operación que permite cambiar los elementos de los distintos sistemas de la arquitectura. Una acción evolutiva debe verificar un conjunto de restricciones para garantizar la consistencia del sistema modificado, y puede implicar la propagación del cambio dentro y fuera de éste.

3.1 El Sistema de Memorización

El Sistema de Memorización almacena, estructura y mantiene el dominio conceptual y de información del sistema hipermedia. Dicho de otro modo, es el responsable de estructurar semánticamente el conocimiento, y de organizar el conjunto de ítems de información, catalogándolos a través de los dominios conceptuales. Siguiendo [García, 01c] el Sistema de Memorización se puede definir como sigue:

> **Def 6.6 [Sistema de Memorización]** El Sistema de Memorización es uno de los cuatro sistemas del modelo SEM-HP. Establece la materia prima con la que se ha de construir el sistema hipermedia.

El modelo de representación utilizado para describir ambos dominios, conceptual y de información, es una estructura conceptual. La estructura conceptual es un grafo dirigido débilmente conectado, que incluye dos tipos de nodos para representar conceptos e ítems de información. Es una red semántica, puesto que tanto sus nodos como sus arcos están etiquetados semánticamente.

> **Def 6.7 [Estructura conceptual]** Una estructura conceptual se define formalmente mediante una tupla:
>
> $$EC = (C, I, Rc, Rf, Ac, Af)$$
>
> en la que *C* es un conjunto de conceptos, *I* es un conjunto de ítems de información, *Rc* es un conjunto de relaciones conceptuales, *Rf* es un conjunto de relaciones

funcionales, *Ac* es un conjunto de asociaciones conceptuales y Af es un conjunto de asociaciones funcionales.

Una **relación conceptual** $r_c \in Rc$, es la etiqueta del arco que conecta dos conceptos, $c_o \in C$ y $c_d \in C$, en la estructura conceptual.

Una **asociación conceptual** $a_c \in Ac$, está formada por dos conceptos y la relación conceptual r_c existente entre ellos $<c_o, r_c, c_d>$.

Una **relación funcional** $r_f \in Rf$, etiqueta el enlace que asocia un ítem, $i_j \in I$, al concepto que lo identifica, $c_k \in C$, y representa el rol o función que la información del ítem desempeña respecto al concepto.

Una **asociación funcional** $a_f \in Af$, está formada por un concepto, un ítem y la relación funcional existente entre ellos $<c_k, r_f, i_j>$.

Cada ítem tiene asociadas una serie de propiedades que describen el tipo y la funcionalidad (rol) de la información que contiene. El nombre y significado de cada propiedad se muestra en la tabla 1.

Tabla 1: Propiedades de un ítem de información

Propiedad	Descripción
Autor	Información acerca del creador del contenido del ítem.
Medio	Tipo de "lenguaje" en que se expresa el contenido del ítem: texto, sonido, gráfico, imagen, animación, aplicación, etc.
Idioma	Lengua en la que se presenta la información del ítem.
Fecha	Fecha de la última actualización del contenido del ítem.
Nivel de dificultad	Grado de especialización del lector al que va dirigida. De manera simplista, permite determinar si la información presentada en el ítem es compleja o sencilla.
Rol	Papel o función que el ítem puede desempeñar en el contexto del sistema de información. Posibles roles son: introducción, definición, ejemplo, aclaración, bibliografía, algoritmo, profundización, recomendación, opinión, antecedente, comparación, etc.

La estructura conceptual contenida en el Sistema de Memorización recibe el nombre de **estructura conceptual de memorización**, EC_M, y puede definirse como sigue:

Def 6.8 [EC_M] La estructura conceptual de memorización es una tupla $EC_M = (C, I, Rc, Rf, Ac, Af)$, que representa íntegramente a través de sus elementos el dominio conceptual y de información del sistema hipermedia.

El *dominio conceptual* es representado en la estructura conceptual de memorización a través del conjunto de conceptos (C) y las asociaciones conceptuales definidas entre ellos (Ac). El *dominio de información* está formado por el conjunto de ítems de información (I) y las asociaciones funcionales (Af) que los ligan a los conceptos. A partir de ambos dominios podemos definir el dominio de conocimiento como sigue:

Def 6.9 [Dominio de conocimiento] El dominio de conocimiento del sistema hipermedia se representa en la estructura conceptual de memorización, y constituye

la unión de un dominio conceptual y un dominio de información definido sobre éste.

La unión de ambos dominios se realiza de modo que cada ítem esté asociado a algún concepto del dominio conceptual. Esto obliga a que, en la fase de navegación, los enlaces entre los ítems se establezcan teniendo en cuenta las relaciones semánticas entre los conceptos que los identifican y no de forma caprichosa o aleatoria.

La figura 4 muestra una estructura conceptual de memorización cuyo dominio de conocimiento es un tipo de Yoga conocido como "Hatha-Yoga". Obviamente, es una estructura de ejemplo y no es completa puesto que existen muchos conceptos que no han sido representados.

El objetivo del ejemplo es mostrar el aspecto que visualmente presenta una estructura conceptual. Con la representación adoptada, los conceptos se dibujan como círculos o elipses, las relaciones conceptuales son flechas, los ítems de información tienen forma rectangular y las relaciones funcionales se dibujan como líneas.

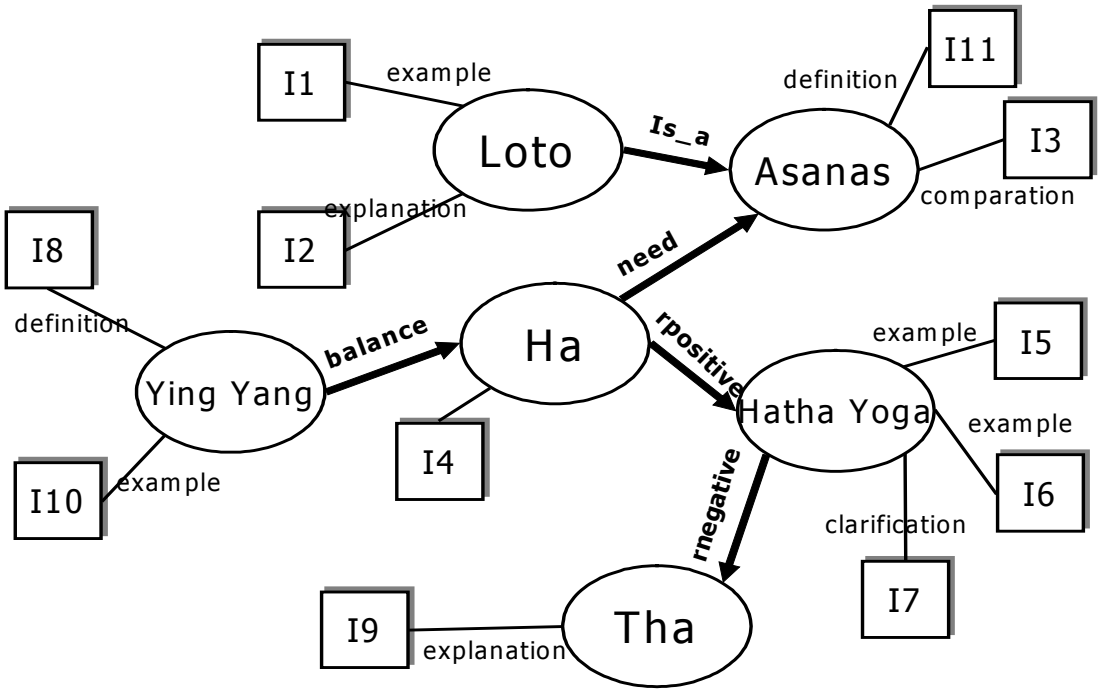


Figura 4: Estructura Conceptual de Memorización

El enfoque evolutivo de SEM-HP permite en cualquier momento que el autor amplíe, reduzca o modifique la estructura conceptual de memorización. Para ello existe un conjunto de acciones evolutivas que permiten crear, modificar o borrar un concepto, un ítem, una asociación funcional o una asociación conceptual.

3.2 El Sistema de Presentación

En la fase de presentación el autor selecciona distintos subconjuntos de la estructura conceptual creada en la fase anterior, con objeto de reducir el tamaño y la complejidad de ésta. De este modo, crea un conjunto de posibles vistas o presentaciones de la

estructura conceptual de memorización. Y, de alguna manera, establece los subdominios en que se divide el dominio de conocimiento capturado en el sistema.

Finalmente, en el Sistema de Presentación se almacenan varias presentaciones basadas en un mismo dominio de conocimiento. Se persigue con ello un doble efecto, por un lado proporcionar al usuario una estructura de navegación centrada en la parcela de conocimiento que le interesa, y por otro, reducir los problemas de desorientación propios de la navegación sobre estructuras demasiado extensas.

> **Def 6.10 [Sistema de Presentación]** El Sistema de Presentación es uno de los cuatro sistemas del modelo SEM-HP. Permite seleccionar una parcela de conocimiento, con el fin de presentar una estructura de navegación de tamaño reducido.

Cada una de las presentaciones recibe el nombre de **estructura conceptual de presentación**, de forma abreviada EC_P. Una EC_P contiene un subconjunto de los conceptos, ítems, asociaciones funcionales y asociaciones conceptuales incluidas en la EC_M.

> **Def 6.11 [EC_P]** Una estructura conceptual de presentación se define mediante una tupla $EC_P = (C^p, I^p, Rc^p, Rf^p, Ac^p, Af^p)$ que contiene un subconjunto de los conceptos (C), ítems (I), relaciones conceptuales (Rc), relaciones funcionales (Rf), asociaciones conceptuales (Ac) y asociaciones funcionales (Af) presentes en la estructura conceptual de memorización.

En la figura 5 se muestra cómo a partir de una EC_M se pueden definir *n* presentaciones diferentes: EC_P^1, EC_P^2,..., EC_P^n. Cada estructura de presentación se define mediante una tupla de EC (Def 6.7), donde cada elemento es un subconjunto del elemento que ocupa la misma posición en la tupla que define la estructura de memorización de la que deriva.

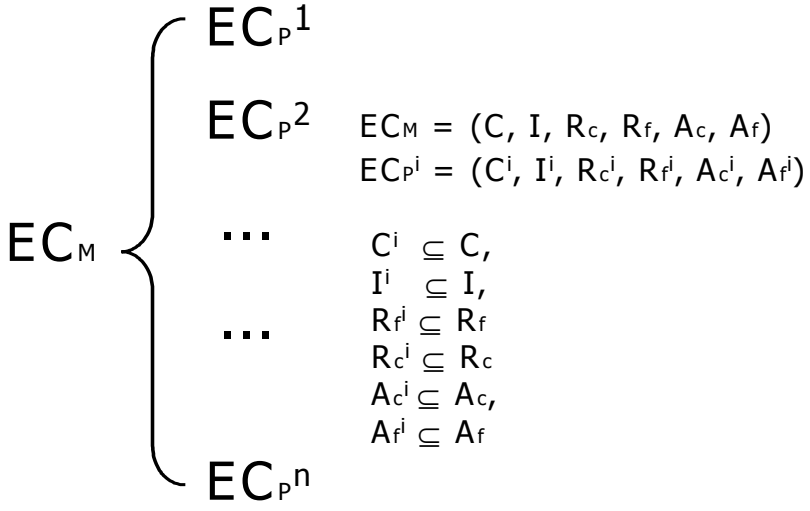


Figura 5: Presentaciones de una EC_M

El subconjunto seleccionado en una presentación debe constituir una subestructura conceptual, por lo tanto debe cumplir una serie de propiedades, como por ejemplo que todos los conceptos elegidos estén conectados.

Por otro lado, al crear una EC_P el autor captura uno o varios subdominos del dominio de conocimiento completo. El autor es el encargado de definir los subdominios (nombre y descripción) que existen e identificar los sudominios de conocimiento representados en cada EC_P. Un mismo subdominio puede ser integrado total o parcialmente en varias presentaciones, por eso es importante indicar para cada subdominio el porcentaje aproximado que se recoge en la presentación.

La figura 6 muestra dos posibles presentaciones creadas a partir de la estructura conceptual del ejemplo de la figura 4. Como puede observarse, en ninguna de las dos aparecen conceptos desconectados.

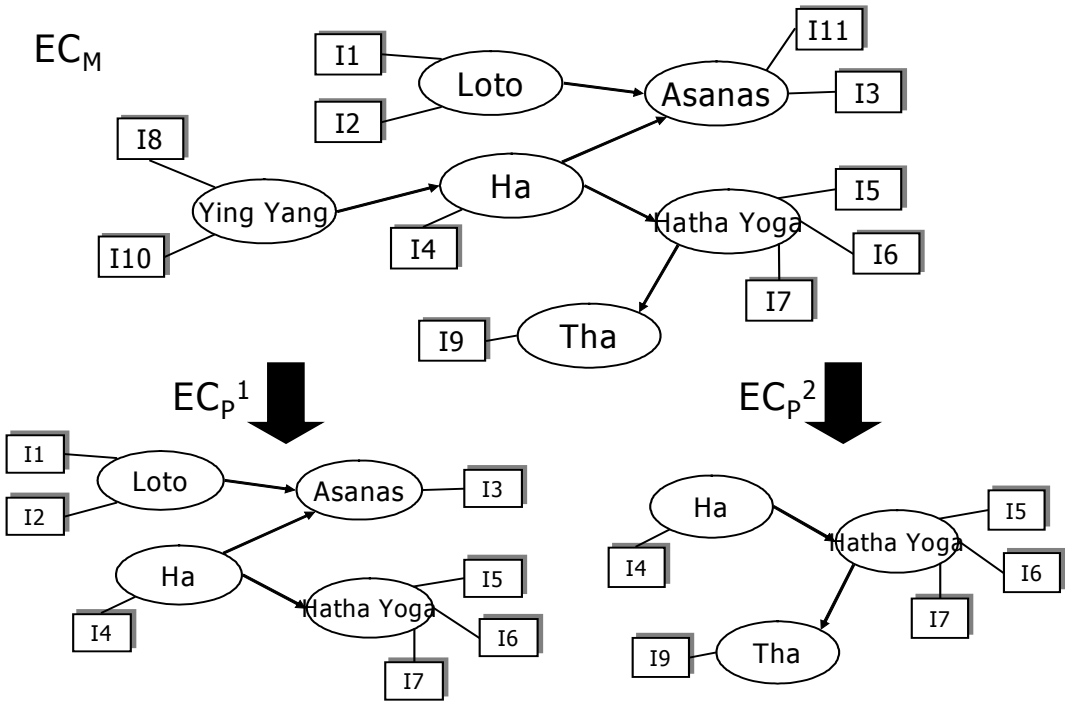


Figura 6: Dos presentaciones diferentes de la misma EC_M

Después de y durante la creación de una estructura conceptual de presentación, el autor puede ocultar o mostrar: conceptos, ítems, asociaciones conceptuales y asociaciones funcionales de la estructura conceptual completa. Para ello, debe hacer uso de las acciones evolutivas asociadas al Sistema de Presentación, de cuya correcta y consistente aplicación se encarga el meta-sistema.

3.3 El Sistema de Navegación

El Sistema de Navegación permite ordenar de algún modo la forma en que se navegan las estructuras conceptuales facilitadas desde el Sistema de Presentación.

> **Def 6.12 [Sistema de Navegación]** El Sistema de Navegación es uno de los cuatro sistemas del modelo SEM-HP. Permite restringir la navegación de una presentación en un orden coherente con las relaciones conceptuales incluidas en ésta.

Por defecto, se define un orden parcial para recorrer una estructura conceptual de presentación. Este orden se basa en:

1) El último ítem de información visitado por el usuario, es decir, desde el que pide saltar a otro ítem,

2) El concepto al que se asocia funcionalmente dicho ítem, y

3) Los conceptos a los que se puede llegar desde ese concepto siguiendo una relación conceptual.

Las restricciones que el autor puede especificar en el Sistema de Navegación son de dos tipos: de navegabilidad y de orden. Las **restricciones de navegabilidad**, RTnb, determinan en que sentido es navegable una relación conceptual. Por defecto, las relaciones conceptuales serán navegadas desde el concepto origen hasta el concepto destino. Sin embargo, si el autor lo desea, puede ampliar la navegabilidad de una relación conceptual en los dos sentidos.

Las restricciones o **reglas de orden**, Ro, establecen si desde un ítem se puede ir hacia otro, haciendo depender esta navegación de las relaciones conceptuales que el lector puede seguir a continuación y los ítems que ha visitado anteriormente. Por defecto, para visitar un ítem asociado a un concepto c_d, el sistema únicamente exige haber visitado previamente alguno de los ítems asociados a un concepto c_o, desde el que parte una relación conceptual navegable hacia c_d. No obstante, en la regla de orden asociada a un ítem, el autor puede exigir además una visita anterior a todos aquellos ítems que, a su juicio, es necesario inspeccionar antes que el actual.

Para cada estructura conceptual creada en la fase de presentación, el autor puede definir varias posibilidades de navegación. Existiendo siempre, como mínimo, una navegación para cada EC_P, que contiene el conjunto de restricciones de navegabilidad y las reglas de orden generadas por defecto.

En cualquier caso, una **estructura conceptual de navegación**, EC_N, queda perfectamente identificada por la estructura conceptual de presentación para la que se define y las restricciones de navegación impuestas sobre ésta.

Def 6.13 [EC_N] Una estructura conceptual de navegación se define con una tupla $EC_N = (EC_P^i, RTnb, Ro)$, donde el primer elemento representa una estructura conceptual de presentación, y los dos últimos un conjunto de restricciones de navegabilidad y reglas de orden que limitan parcialmente la forma en que ésta va a poder recorrerse.

3.4 El Sistema de Aprendizaje

El Sistema de Aprendizaje es el que permite calificar de adaptativos a los sistemas hipermedia desarrollados de acuerdo al modelo SEM-HP. Este sistema se propone, pero no se aborda en la tesis realizada por la doctora García-Cabrera [García, 01c]. Y, como ya se expuso en el prefacio, constituye el tema central de la presente tesis. En esta sección ofrecemos un breve resumen que se amplía, detalla y formaliza en posteriores capítulos.

Def 6.14 [Sistema de Aprendizaje] El Sistema de Aprendizaje es uno de los cuatro sistemas del modelo SEM-HP. Se encarga de modelar al usuario y adaptar

la estructura y el funcionamiento del sistema hipermedia a sus características personales.

Para conseguir que la adaptación al usuario sea lo más completa posible, la información reunida en el **modelo de usuario**, MU, es de muy diversa índole: desde datos personales hasta el estado de conocimiento que el usuario posee sobre los ítems y conceptos del sistema, pasando por su experiencia en la materia, experiencia en navegación hipermedia, preferencias, metas e intereses.

Para llevar a cabo el proceso de adaptación, el Sistema de Aprendizaje aplica una serie de técnicas y métodos adaptativos que se basan en tres conjuntos de reglas definidas previamente por el autor [Medina, 02c].

- **Reglas de actualización**, Ru: Incrementan el grado de conocimiento que el usuario posee sobre los ítems de información a medida que éste los visita.

- **Reglas de peso**, Rw: Calculan el conocimiento del usuario acerca de un concepto usando el grado de conocimiento que éste posee sobre cada ítem asociado funcionalmente al concepto.

- **Reglas de conocimiento**, Rk: Determinan en cada momento qué ítems puede visitar el usuario y para cuales no está capacitado. Además, indican cuáles de los ítems accesibles contienen información relevante dado su actual estado de conocimiento.

Parte de la estructura del modelo de usuario y las reglas de peso dependen de los conceptos, ítems y asociaciones funcionales incluidas en la estructura conceptual de memorización, por lo tanto se definen de forma unívoca para ésta. Respecto a las reglas de actualización y conocimiento, el autor puede definir distintos conjuntos para una misma estructura conceptual de navegación (figura 7). Existiendo, en cualquier caso, un conjunto de reglas de actualización y conocimiento por defecto para cada EC_N.

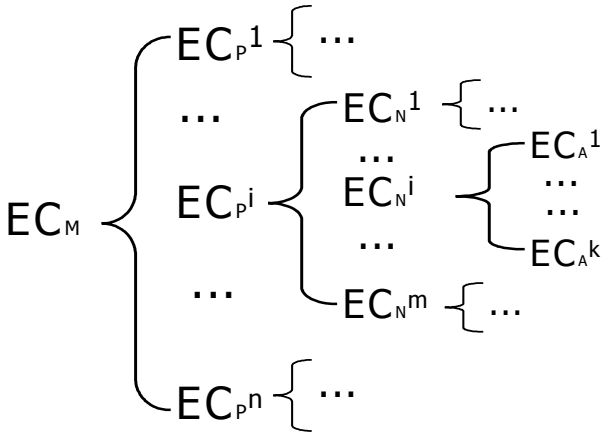


Figura 7: Árbol de estructuras conceptuales

Def 6.15 [EC_A] Una estructura conceptual de aprendizaje queda perfectamente definida a partir de una tupla $EC_A = (EC_M, Rw, MU, EC_N^j, Ru, Rk)$, de forma extendida $EC_A = (EC_M, Rw, MU, EC_P^i, RTnb^j, Ro^j, Ru, Rk)$, donde se establece un conjunto de reglas de peso y un modelo de usuario para la EC_M, y se define un conjunto de reglas de orden, de navegabilidad, de conocimiento y de actualización para una de sus presentaciones, EC_P^i.

Como se verá más detalladamente, para cada usuario se elige de forma personalizada una estructura conceptual de aprendizaje. Esta selección se realiza evaluando cada estructura, de acuerdo a la experiencia e intereses del usuario, y optando por la que mejor se le ajusta.

Los **métodos de adaptación** aplicados están destinados en su mayoría a *personalizar la navegación* del usuario, modificando y enriqueciendo la estructura de navegación que se le proporciona; con el fin último de reducir los problemas de desorientación y falta de comprensión que pueden surgirle durante su proceso de navegación. Entre otras, el sistema ejecuta técnicas para ocultar y deshabilitar los ítems inaccesibles, anotar los ítems interesantes, generar rutas guiadas que permiten alcanzar una meta de conocimiento específica, reflejar el estado de conocimiento del usuario sobre la propia estructura de navegación, etc. También se aplican métodos para *personalizar la presentación* de la información proporcionada, por ejemplo a través de la composición de ítems.

Además, el sistema permite al usuario recorrer la información de cuatro formas distintas. Cada una de estas modalidades está sujeta a un conjunto de restricciones diferentes. De forma que las posibilidades de elección de un usuario serán distintas en cada una de ellas.

- **Navegación tradicional**: Se corresponde con el tipo de navegación a la que los usuarios de sistemas web están acostumbrados. No existe ninguna restricción de acceso, de modo que en cualquier momento un ítem puede ser explorado.

- **Navegación por conceptos**: Es también una navegación libre, donde el usuario recorre conceptos en lugar de ítems. La selección de un concepto genera un documento que integra los ítems de información asociados a él.

- **Navegación por relación conceptual**: Es obligatorio recorrer la información en un orden coherente con las relaciones existentes entre los conceptos. La selección de un ítem está permitida si así se establece en las reglas de orden especificadas en el Sistema de Navegación.

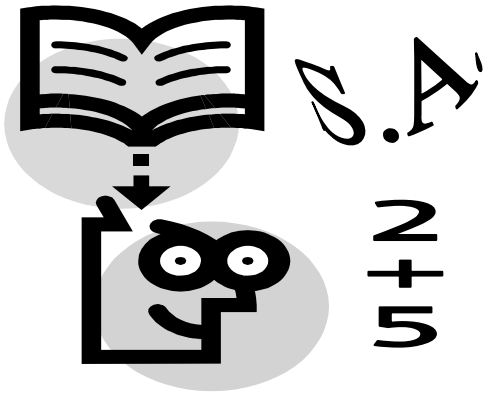
- **Navegación restringida por conocimiento**: Únicamente se puede acceder al contenido de un ítem si se tiene el conocimiento previo necesario para ello. En este caso, las restricciones de navegación son las que se definen en las reglas de conocimiento.

El usuario puede optar por el modo de navegación que más le convenga o apetezca en cada momento. Sea cual sea el modo escogido, el sistema actualiza el conocimiento del usuario, siempre y cuando en el momento de la visita el usuario satisfaga las restricciones pedagógicas impuestas por el autor en las reglas de conocimiento.

Además, a partir del proceso de navegación del grupo de usuarios, el Sistema de Aprendizaje es capaz de sacar ciertas conclusiones, que pueden ayudar al autor a detectar diferencias entre sus modelos de representación y la concepción que de éstos tienen quienes los navegan. Dicho de otra forma, el Sistema de Aprendizaje infiere una serie de modificaciones sobre el resto de sistemas, principalmente sobre el de memorización, que el autor puede llevar a cabo para ajustar, aún más, las estructuras de navegación a sus usuarios.

CAPÍTULO 7

El Sistema de Aprendizaje



Resumen

El presente capítulo tiene como objetivo presentar el Sistema de Aprendizaje, introduciendo conceptualmente los elementos que lo constituyen, siempre desde una perspectiva informal. Se resumen los mecanismos proporcionados al autor para establecer requisitos pedagógicos entre los ítems de información, y para modelar la adquisición de conocimiento del usuario. Se desglosan los atributos recogidos en el modelo del usuario y su utilidad durante el proceso de adaptación. Se esboza el proceso seguido para elegir de forma personalizada la estructura de navegación, y se describen los cuatros modos en que ésta puede ser recorrida. Se indican las técnicas de adaptación propias de cada modo, destacando la adaptación llevada a cabo durante la navegación por conocimiento. Finalmente se describe la integración del Sistema de Aprendizaje dentro del modelo SEM-HP, desde un punto de vista evolutivo y adaptativo.

Tabla de contenidos

El Sistema de Aprendizaje

1. INTRODUCCIÓN

Como ya se indicó en el capítulo anterior, el objetivo de la presente tesis es definir y formalizar un modelo de adaptación que, sin dejar de tener una aplicación general, permita su completa integración en el modelo SEM-HP, concretamente en el Sistema de Aprendizaje. En este capítulo se describen, a nivel conceptual, los distintos elementos que constituyen el Sistema de Aprendizaje, resaltando en cada caso las ventajas que se incorporan, tanto desde la perspectiva del autor como del usuario. También se aborda, de manera muy general, la interrelación existente entre el Sistema de Aprendizaje y el resto de sistemas que constituyen SEM-HP.

A lo largo del capítulo se van presentando los diferentes elementos, tanto funcionales como estructurales que componen el Sistema de Aprendizaje. Cada uno de estos elementos es especificado, descrito y formalizado en un capítulo posterior. La distribución de dichos capítulos puede ser consultada en la sección 5.

Para facilitar su comprensión, durante el desarrollo del presente capítulo se recurre frecuentemente a la exposición de ejemplos. En los que, sin utilizar ningún formalismo, esto es, usando únicamente el lenguaje natural, se pretende poner de manifiesto la estructura y funcionalidad del modelo de adaptación.

Para ello, se parte de la estructura conceptual de memorización (ECM) mostrada en la figura 1, cuyo dominio de conocimiento es el paradigma de orientación a objetos.

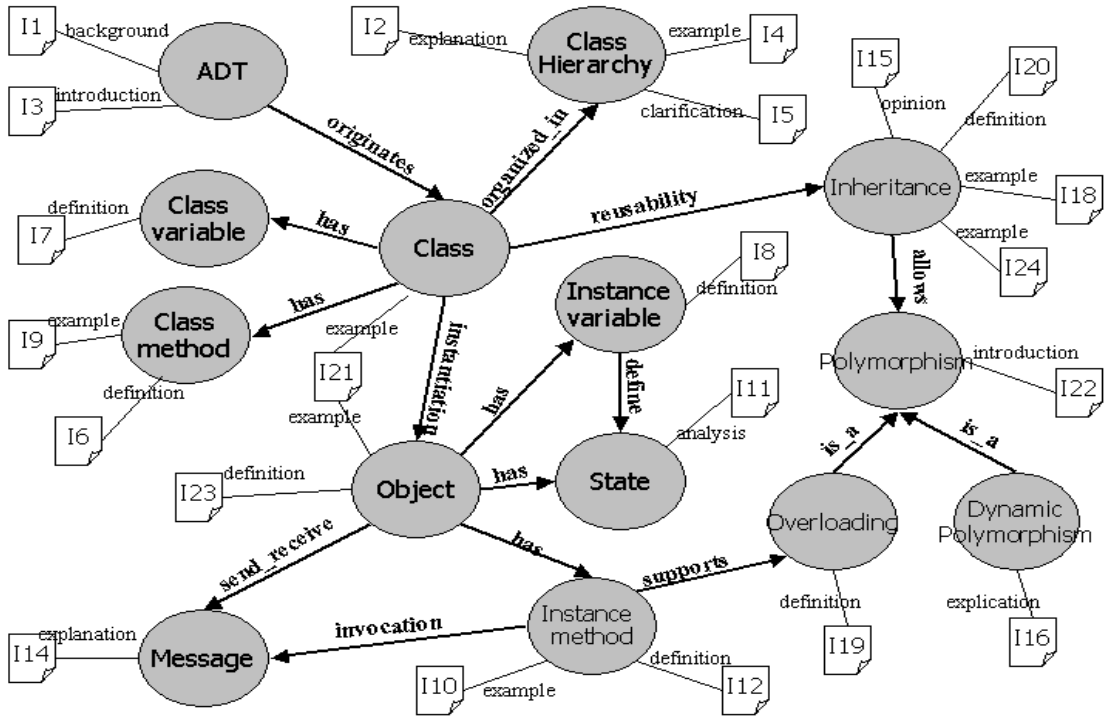


Figura 1: Estructura Conceptual de Ejemplo

Como puede verse en la figura, el conjunto de conceptos está formado por las ideas primordiales de la orientación a objetos,

$$C = \{Clase, Objeto, TDA, Herencia, Mensaje, ...\}$$

y los ítems, I, son trozos de información que tratan sobre dichos conceptos. Por ejemplo, el ítem I15 expresa una opinión referente al uso de la herencia múltiple y el ítem I10 contiene ejemplos de métodos de instancia en distintos lenguajes de programación.

El conjunto de relaciones conceptuales (Rc) utilizadas para etiquetar las asociaciones entre conceptos incluye entre otras las siguientes: "originates", "send_receive", "organized_in", "invocation", cuya semántica es definida en relación al paradigma de orientación a objetos.

Finalmente, las relaciones funcionales (Rf) utilizadas para definir la asociación entre un ítem y un concepto son: "background", "introduction", "definition", "example", "explanation", etc.

2. El Sistema de Aprendizaje para el Autor

Sin la existencia del Sistema de Aprendizaje, el modelo SEM-HP ofrece al usuario una vista de la estructura conceptual de memorización, representada también mediante una red semántica, y denominada estructura conceptual de presentación (ECP). Los posibles recorridos que el usuario puede realizar sobre esta estructura son limitados mediante restricciones de orden (Ro) y navegabilidad (RTnb). De tal manera que, como se explicó en el capítulo anterior, la estructura proporcionada al usuario desde el Sistema de Navegación se compone de los siguientes elementos:

$$EC_N = (EC_P^i, Ro, RTnb) \qquad (1)$$

Con los medios anteriores, la navegación sólo puede restringirse en función de las visitas que realiza el usuario en el sistema, ignorando información adicional como su conocimiento, preferencias o intereses. Justamente, el Sistema de Aprendizaje proporciona los instrumentos necesarios para tener en cuenta esas y otras características del usuario, permitiendo posteriormente adaptar convenientemente tanto la estructura como el proceso de navegación.

2.1 Involucrar el conocimiento del usuario en la navegación

Una de las principales ventajas que incorpora el Sistema de Aprendizaje es la posibilidad de dirigir la navegación del usuario de acuerdo a su estado de conocimiento. El instrumento proporcionado para ello es denominado, de forma colectiva, **reglas de conocimiento** (Rk).

El autor define reglas de conocimiento para decir qué cosas debe conocer el usuario y a qué nivel, con la finalidad de asegurar que éste asimila correctamente el contenido de un determinado ítem de información. También puede establecer si un ítem contiene información relevante para el usuario, en función del conocimiento que éste posee sobre

otros ítems relacionados. Es decir, puede que no tenga sentido invertir tiempo en leer un ítem si ya se conocen suficientemente otros ítems que tratan la misma información.

Por ejemplo, en la EC_P de la figura 2, el autor puede determinar que para comprender el ejemplo de *Class Method* incluido en el ítem I9 es necesario:

a) tener un conocimiento alto sobre la noción *Instance Method* (definición en I12 y ejemplo en I10) y tener más o menos clara la definición de *Object* (en I23), o bien,

b) haber estudiado en profundidad un ejemplo de *Class* (en I21) y conocer superficialmente la definición de *Class Method* (en I6).

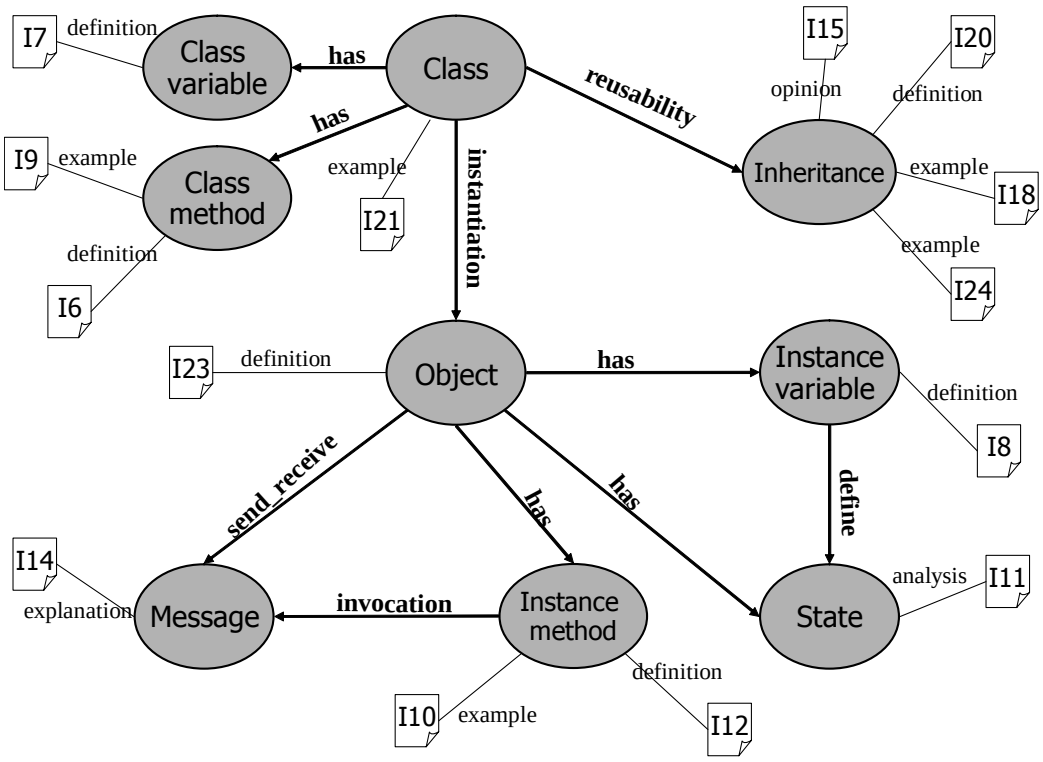


Figura 2: Una posible presentación

El primer problema que se plantea es decidir el modo de representar los distintos grados o niveles de conocimiento, por ejemplo: ¿Cómo expresamos que el conocimiento es alto?. Una posibilidad interesante es representar el conocimiento como un valor porcentual, desde 0 hasta 100 [DeBra, 98b]. Sin embargo, los autores no suelen ser tan precisos cuando hablan de conocimiento y les puede resultar complicado y artificioso asignar un número exacto a la concepción de bastante, poco, mucho, nada, etc. Por este motivo pediremos al autor que exprese los niveles de conocimiento usando etiquetas semánticas. Y para que tenga suficiente expresividad, además de las etiquetas “nulo”, “medio” y “total”, permitimos dos valores intermedios: “bajo” y “alto”.

Utilizando esas etiquetas semánticas, y de manera intuitiva e informal, los requisitos o condiciones de conocimiento establecidos para I9 en el ejemplo anterior pueden ser expresados en dos reglas de conocimiento, tal y como se muestra en la tabla 1.

Tabla 1: Reglas de conocimiento para I9

1	I9 es accesible si el conocimiento sobre I12 es mayor o igual que ”alto”, el conocimiento sobre I10 es mayor o igual que ”alto” y el conocimiento sobre I23 es mayor o igual que “medio”.
2	I9 es accesible si el conocimiento sobre I21 es igual a “total” y el conocimiento sobre I6 es mayor o igual que “bajo”.

A este tipo de restricciones, que exigen un conocimiento mínimo sobre un ítem o conjunto de ítems, las denominamos **restricciones de accesibilidad** ya que determinan las circunstancias en que el ítem para el que se imponen puede ser accedido. De este modo, siguiendo el ejemplo, un usuario que no satisface al menos una de las dos reglas de la tabla 1 no podría acceder al contenido de I9.

También con las reglas de conocimiento (véase tabla 2), el autor puede expresar cosas del tipo:

Si el usuario ha conseguido un conocimiento alto sobre el ítem I18, el cual presenta un ejemplo complejo de *Inheritance*, no es relevante que visite I24, puesto que éste contiene una versión simplificada de dicho ejemplo.

Estas **restricciones** establecen la **idoneidad** de la visita al ítem sobre el que se imponen, exigiendo un grado de conocimiento máximo sobre determinados ítems.

Tabla 2: Regla de conocimiento para I24

I24 es idóneo si el conocimiento sobre I18 es menor o igual que “alto”.

Por supuesto en una misma regla de conocimiento se pueden combinar restricciones de idoneidad y de accesibilidad. El autor puede establecer varias reglas de conocimiento para un ítem. En este caso, tal y como se mencionó para el ejemplo de la tabla 1, es suficiente con que el usuario satisfaga una de ellas. Por supuesto, el autor sólo crea reglas de conocimiento para los ítems que a su juicio es necesario, no siendo obligatoria la existencia una regla de conocimiento para cada ítem.

2.2 Obtener el conocimiento del usuario

Si el autor desea tutelar la navegación del usuario en función del conocimiento que éste posee, es necesario definir un mecanismo que permita actualizar su estado de conocimiento a medida que lee información en el sistema hipermedia. Este mecanismo debe ser capaz de obtener el nuevo estado de conocimiento del usuario, sin necesidad de preguntarle explícitamente en cada paso. Con este fin, el sistema incorpora un nuevo instrumento denominado **reglas de actualización** (Ru).

En esta ocasión, sí es necesario establecer una regla de actualización para cada ítem de la EC_P. Concretamente, sólo se permite una regla para cada ítem. Esta regla establece cómo cambia el conocimiento del usuario después de visitar el ítem sobre el que se define. Para facilitar la tarea del autor el sistema genera de forma automática una regla de actualización para cada ítem. Esta regla por defecto fija a “total” el conocimiento del usuario después de leer el ítem asociado (véase la tabla 3).

Tabla 3: Regla de actualización por defecto para Ij

Después de visitar Ij, el conocimiento sobre Ij se fija a "total".

No obstante, el autor puede modificar la regla de actualización de un ítem para expresar otro tipo de actualización sobre el ítem visitado e incluso actualizaciones sobre otros ítems relacionados.

Por ejemplo, usando las reglas de actualización (véase la tabla 4), el autor puede expresar cosas del tipo:

Debido a la complejidad del ejemplo de *Inheritance* presentado en I18, el usuario debe leer, al menos, dos veces su contenido para comprenderlo completamente.

La primera vez que el usuario consulta la definición de *Instance Method* incluida en I12, aprende un poco más acerca de la definición de *Class Method* establecida en I6.

Debido a la estrecha relación que existe entre el contenido de los ítems I12 e I10 (respectivamente definición y ejemplo de *Instance Method*), el conocimiento del usuario sobre I10 debe ser igual a su conocimiento acerca de I12.

Tabla 4: Dos reglas de actualización

a)	Después de visitar I8, el conocimiento sobre I8 se incrementa en dos grados de conocimiento.
b) y c)	Después de visitar I12, el conocimiento sobre I12 se fija a "total", el conocimiento sobre I6 se incrementa en un grado de conocimiento siempre y cuando sea la primera visita a I12 y, además, el conocimiento sobre I10 se fija al conocimiento alcanzado sobre I12.

Como se verá con más detalle en el capítulo destinado a las reglas de actualización, se permite al autor establecer con gran expresividad los cambios de conocimiento que acontecen al usuario a medida que éste navega. La *actualización* de un ítem puede ser *absoluta o incremental*. Además, se permite *hacer referencia al conocimiento del ítem visitado* en la actualización de otros ítems incluidos en la regla. Al mismo tiempo que una actualización puede ser *ejecutada cada vez que se visita el ítem o sólo después de la primera visita*.

En cualquier caso, las reglas de actualización nunca disminuyen el estado de conocimiento del usuario. Es decir, cuando el usuario lee la información contenida en un ítem, mantiene o aumenta su conocimiento actual, pero nunca lo pierde o mengua. Por este motivo una actualización sobre un ítem no se aplica si supone un grado de conocimiento menor que el que actualmente posee el usuario.

Gracias a las reglas de actualización, el Sistema de Aprendizaje es capaz de obtener el grado de conocimiento que el usuario posee acerca de los distintos ítems de información ofrecidos en el sistema hipermedia. La utilización de este conocimiento es esencial para realizar el proceso de adaptación sobre el hipermedia. Pero, además, puede ser utilizado[1] para calcular el conocimiento del usuario acerca de los conceptos del sistema.

[1] Recordemos que cada concepto puede tener asociados distintos ítems de información.

El instrumento que utilizamos para extraer este conocimiento conceptual a partir del conocimiento que el usuario posee sobre los ítems se denomina **reglas de peso** (Rw). Desde un punto de vista pedagógico es importante, ya que permite al usuario conocer su grado de conocimiento acerca de los conceptos que sustentan el sistema hipermedia y no simplemente sobre los trozos de información que "explican" parte de dichos conceptos.

Existe una regla de peso para cada concepto incluido en la EC_M. Esta regla de peso involucra a todos los ítems asociados funcionalmente a dicho concepto. De nuevo, con la intención de simplificar la tarea del autor, el sistema genera una regla de peso para cada concepto. Esta regla por defecto, obtiene el conocimiento del usuario acerca de un concepto como el grado de conocimiento medio que el usuario posee sobre cada uno de los ítems ligados a él en la EC_M. En la tabla 5 se muestra la regla de peso generada por defecto para el concepto *Inheritance*.

Tabla 5: Regla de peso por defecto para *Inheritance*

El conocimiento sobre *Inheritance* es igual a: ¼ del conocimiento sobre I15 más ¼ del conocimiento sobre I18 más ¼ del conocimiento sobre I20 más ¼ del conocimiento sobre I24.

Sin embargo, el autor puede considerar que determinados ítems contienen información mucho más significativa que otros para comprender el concepto.

En nuestro ejemplo, el autor piensa que la definición de *Inheritance*, en I20, es más importante que los ejemplos, I18 e I24, asociados a este concepto, y que la opinión recogida en I15 es poco representativa para captar el significado del concepto.

Para representar esta diferencia de importancia, el autor puede modificar la distribución de pesos por defecto, por ejemplo como se muestra en la tabla 6. En cualquier caso la distribución de pesos siempre debe sumar 1.

Tabla 6: Regla de peso para *Inheritance*

El conocimiento sobre *Inheritance* es igual a: $\frac{1}{8}$ del conocimiento sobre I15 más $\frac{2}{8}$ del conocimiento sobre I18 más $\frac{3}{8}$ del conocimiento sobre I20 más $\frac{2}{8}$ del conocimiento sobre I24.

2.3 La evolución de las reglas

El Sistema de Aprendizaje, al igual que el resto de sistemas existentes en SEM-HP, incorpora un conjunto de **acciones evolutivas**, que pueden ser utilizadas por el autor con el objeto de modificar cualquiera de los tres conjuntos de reglas ilustradas en las secciones anteriores.

Cada acción evolutiva implica un tipo de cambio y tiene asociadas una serie de **restricciones**, que deben satisfacerse completamente para que el cambio sea efectivo. Algunas de estas restricciones podrán comprobarse a priori (*pre-condiciones*) mientras que para validar otras será necesario simular el estado tras el cambio (*post-condiciones*). El meta-sistema es nuevamente el encargado de mantener, validar, y ejecutar las acciones evolutivas sobre los distintos elementos del Sistema de Aprendizaje.

El conjunto de acciones evolutivas asociadas a cada elemento del Sistema de Aprendizaje es descrito de manera exhaustiva en el capítulo que define, detalla y formaliza dicho elemento. En esta sección simplemente mostramos las principales modificaciones que estas acciones permiten realizar sobre las reglas de conocimiento, actualización y peso. Resaltando, además, determinadas restricciones que garantizan la consistencia de los cambios realizados y el correcto funcionamiento del sistema.

Tabla 7: Evolución de los elementos del Sistema de Aprendizaje

Reglas de Conocimiento	Acciones evolutivas	Crear una nueva regla de conocimiento para un ítem. Eliminar una regla de conocimiento de un ítem. Añadir nuevas restricciones de conocimiento a una regla. Eliminar o modificar las restricciones de conocimiento existentes en una regla.
	Restricciones	Comprobar que las restricciones de conocimiento son factibles, por ejemplo, no es válida una restricción que exige un conocimiento menor que "nulo". Garantizar que las restricciones de conocimiento impuestas en la regla no son contradictorias entre sí. Por ejemplo, no es posible que el conocimiento sobre un ítem sea mayor que "alto" y menor que "medio" al mismo tiempo.
Reglas de Actualización	Acciones evolutivas	Añadir una actualización en la regla de actualización de un ítem. Eliminar o modificar una actualización en la regla de actualización de un ítem.
	Restricciones	El valor de conocimiento actualizado debe estar entre los valores "nulo" y "total". No pueden existir dos actualizaciones diferentes sobre un ítem en una misma regla de actualización. La ejecución repetida, una o más veces, de una regla de actualización debe alcanzar conocimiento "total" sobre el ítem visitado.
Restricciones		El conjunto de reglas de conocimiento y actualización debe garantizar que todos los ítems pueden llegar a ser accesibles en algún momento.
Reglas de Peso	Acciones evolutivas	Modificar la distribución de pesos en una regla de peso. Reestablecer la distribución de pesos por defecto en una regla de peso.
	Restricciones	La distribución de pesos de una regla debe sumar siempre 1.

2.4 La nueva estructura de navegación: EC_A

En ausencia del Sistema de Aprendizaje, el modelo SEM-HP ofrecía al usuario una estructura de navegación, EC_N, compuesta por una estructura conceptual de presentación, EC_P^i, y un conjunto de reglas capaces de restringir parcialmente los posibles recorridos a través de ésta.

El Sistema de Aprendizaje superpone a los anteriores elementos, un conjunto de reglas de peso definido sobre los conceptos de la EC_M y un conjunto de reglas de conocimiento y de actualización que determinan la navegación y adquisición de conocimiento sobre los ítems de la estructura conceptual de presentación.

En consecuencia, la **estructura de navegación** ofrecida al usuario, es ahora, una estructura conceptual de aprendizaje (Def 6.15):

$$EC_A = (EC_M, Rw, MU, EC_P^i, RTnb^j, Ro^j, Ru, Rk) \qquad (2)$$

Tal y como se explicó en el capítulo anterior, el autor, siguiendo el modelo SEM-HP, realiza un desarrollo incremental del sistema hipermedia: En la primera fase, define la estructura conceptual de memorización, a partir de ésta crea distintas presentaciones, sobre cada presentación puede definir varios esquemas de navegación (reglas de orden y navegabilidad) y finalmente, en la fase de aprendizaje, puede establecer de nuevo distintos conjuntos de reglas de conocimiento y actualización para cada estructura de la fase anterior.

De este modo, si por ejemplo, en cada fase el autor define dos estructuras a partir de cada estructura creada en la fase previa, finalmente se obtendrían 2^3 estructuras de navegación distintas.

La diferencia entre todas las estructuras de navegación disponibles, es algo que sólo su autor conoce. Por eso, es importante que el autor etiquete cada EC_A con información significativa acerca de su contenido y el perfil de usuario al que está dirigida. Este etiquetado permite posteriormente elegir de entre todas las estructuras de navegación existentes aquella que mejor se ajuste a la experiencia e intereses del usuario.

El **etiquetado de una EC_A** no debe suponer demasiado esfuerzo para el autor, así pues, basta con que recoja la siguiente información:

El conjunto de *subdominios de conocimiento que se capturan* en la EC_P^i, indicando para cada subdominio el porcentaje aproximado en que está presente.

Obviamente todas las EC_A definidas sobre una misma presentación coinciden en esta etiqueta. Cada vez que el autor hace referencia a un nuevo subdominio, debe asignarle un nombre que lo identifique. La identificación del subdominio puede hacerse de acuerdo a un criterio personal o consensuado. En cualquier caso, a modo informativo, es conveniente que el autor asocie una breve descripción a cada subdominio.

Grados de *experiencia en la materia* adecuados para navegar y leer sin dificultades de comprensión, ni sensación de excesiva simplicidad, los ítems de información ofrecidos en la estructura de navegación.

Grados de *experiencia en navegación web* adecuados para recorrer sin problemas de desorientación, ni sensación de falta de enlaces, la estructura de navegación.

La figura 3 muestra el etiquetado de dos posibles estructuras conceptuales de aprendizaje, EC_A^1, EC_A^2, creadas a partir de la presentación mostrada en la figura 2.

A partir de las etiquetas es posible deducir que ambas estructuras de navegación están dirigidas a usuarios con poca experiencia en el uso de sistemas web. Y que, mientras la EC_A^1 es adecuada para usuarios con conocimiento en la materia medio-alto, la EC_A^2 está pensada para los menos duchos en el tema.

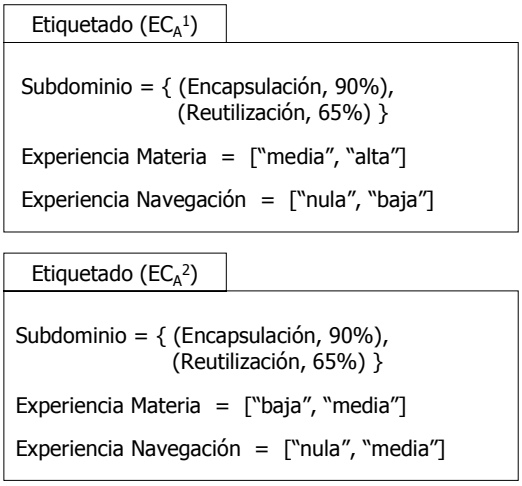


Figura 3: Etiquetado para dos estructuras de navegación

3. EL SISTEMA DE APRENDIZAJE PARA EL USUARIO

Para que la navegación del usuario, y en general, su interacción con el sistema hipermedia sea un proceso personalizado, es necesario que el sistema almacene y consulte alguna información acerca de éste. Como se irá viendo a lo largo del presente capítulo, esta información reúne datos de muy diversa índole: estado de conocimiento, experiencia, intereses, preferencias, etc. Toda esta información se guarda en un **modelo de usuario** (MU) que debe estar actualizado en todo momento para que las adaptaciones realizadas sean realmente beneficiosas.

La estructura del modelo de usuario depende de los ítems y conceptos incluidos en la estructura conceptual de memorización, mientras que el contenido del modelo de usuario es diferente para cada usuario del sistema. El modelo de usuario se crea la primera vez que el usuario se registra en el sistema, y se mantiene a través de todas las sesiones realizadas por éste.

De esta manera, el modelo de usuario contempla su evolución en cuanto a experiencia, conocimientos adquiridos, ítems visitados, etc. Y las adaptaciones se realizan teniendo en cuenta las características del usuario, recopiladas no sólo en la sesión actual, sino a lo largo de todas sus sesiones de trabajo. Para mantener la correspondencia entre el usuario y el modelo de usuario que lo representa dentro del sistema, es necesario que en su primera sesión el usuario se identifique proporcionando algunos datos personales y una clave de acceso.

3.1 Elección de la estructura de navegación

Para comenzar a navegar, el usuario obtiene una estructura navegación, EC_A^k, elegida especialmente para él de entre todas las disponibles. Para poder hacer esta elección, el sistema debe conocer el grado de experiencia (de navegación y en la materia) que el usuario posee. Inicialmente, esta información es solicitada al usuario. Después, en el momento que crea oportuno, el usuario puede modificar directamente esa información o solicitar una actualización automática. En este último caso, el sistema infiere el nuevo

grado de experiencia a partir del número de visitas realizadas y el conocimiento adquirido por el usuario durante su uso del sistema hipermedia.

Otro factor importante para elegir la estructura de navegación es el subdominio o parcela de conocimiento en la que el usuario está interesado. Para ello, se ofrece al usuario una lista con todos los subdominios definidos previamente por el autor, de modo que pueda elegir, en cada momento, el que más deseos tiene de conocer.

Tanto la experiencia como el subdominio de interés forman parte del modelo de usuario y son consultados por el sistema para determinar qué estructura de navegación se le proporciona. Para hacer esta elección, el sistema coteja el etiquetado de cada EC_A con el valor que presentan esos tres atributos en el modelo de usuario.

Primeramente, el sistema evalúa sólo las EC_A que capturan el subdominio que interesa al usuario y que, además, son adecuadas para la experiencia de éste. Para cada una de estas estructuras se obtiene un indicador que expresa el grado en que se ajusta al perfil del usuario. El indicador tiene en cuenta el porcentaje en que el subdominio de interés está incluido en la EC_A y cómo se sitúa el grado de experiencia del usuario dentro del intervalo que etiqueta la estructura. Por supuesto, se elige la estructura de navegación que presenta el mejor indicador.

Por ejemplo, de acuerdo al modelo de usuario mostrado en la figura 4.A, las dos estructuras de navegación de la figura 3 son adecuadas, porque ambas incluyen el subdominio "Reutilización" y el grado de experiencia del usuario está dentro del intervalo establecido, tanto para la experiencia de navegación como en la materia.

Cuando la experiencia del usuario crece de "media" a "alta" (figura 4.B), la estructura de navegación EC_A^2 deja de ser adecuada, eligiéndose por tanto la EC_A^1.

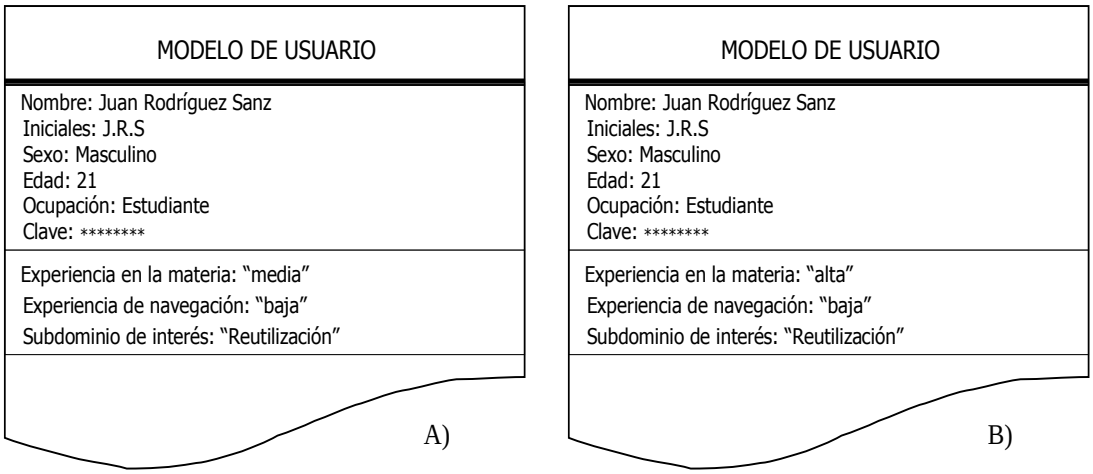


Figura 4: Modelo de usuario para Juan

Puede ocurrir, que el autor no haya creado ninguna EC_A que capture el subdominio deseado y que se ajuste a la experiencia del usuario. En este caso, el sistema plantea la situación al usuario, tras lo cuál, éste puede decidir cambiar su subdominio de interés o mantenerlo. Si elige la segunda opción, el sistema evalúa todas las EC_A definidas sobre ese subdominio y selecciona la que menos se distancie del perfil del usuario.

3.2 Adquisición de conocimiento

En cualquier proceso de aprendizaje la adquisición de conocimiento es un elemento clave. Por este motivo, el Sistema de Aprendizaje incorpora a SEM-HP la capacidad de adaptar la estructura de navegación elegida para el usuario y la manera de recorrerla, en función de su estado de conocimiento actual.

El **estado de conocimiento** se almacena en el modelo de usuario y está formado por el grado de conocimiento que el usuario posee sobre cada ítem presente en el sistema hipermedia. En el modelo de usuario también se contempla el conocimiento del usuario sobre los conceptos del sistema hipermedia, siendo éste un estado de conocimiento derivado del anterior a través de las reglas de peso.

En la sección 2.2 se explicó que el Sistema de Aprendizaje permite al autor definir para cada estructura de navegación, un conjunto de reglas, denominadas de actualización, que estipulan la forma en que el conocimiento del usuario crece a medida que éste lee información en el sistema. Sin embargo, no siempre la lectura de un trozo de información, asegura la adquisición de conocimiento por parte del lector. Salvo en el caso de información muy básica, para transformar información en conocimiento es necesario disponer anteriormente de otro conocimiento. Independientemente de cuál sea este conocimiento previo requerido, parece sensato no actualizar el conocimiento del usuario si no existe garantía de que está preparado para asimilar la información que inspecciona.

Tal y como se expuso en la sección 2.1, el Sistema de Aprendizaje pone al alcance del autor un instrumento lógico, denominado reglas de conocimiento, con el que puede expresar todos los requerimientos de conocimiento que considere necesarios. Concretamente las restricciones de accesibilidad presentes en este tipo de reglas, le sirven al autor para indicar qué conocimiento es necesario que el usuario posea para estar en disposición de interpretar correctamente la información contenida en un ítem.

De este modo, mientras que el usuario navega, cada vez que accede a la información ofrecida en un ítem, el sistema comprueba si su estado de conocimiento actual satisface los requisitos de accesibilidad de alguna de las reglas de conocimiento asociadas a dicho ítem. Sólo si el usuario satisface la anterior condición, o claro está, si no existe ninguna restricción de accesibilidad para el ítem consultado, el sistema acepta como válida la adquisición de conocimiento propuesta por el autor para el ítem visitado. Ejecutando, en consecuencia, las actualizaciones de la correspondiente regla de actualización y modificando el estado de conocimiento del usuario en su modelo.

La figura 5 muestra la actualización del estado de conocimiento de Juan tras visitar los ítems I12 e I8 de la figura 2 para los que el autor no ha impuesto ninguna regla de conocimiento, y cuyas reglas de actualización se muestran en la tabla 4.

Los grados de conocimiento actualizados se han resaltado en negrita. Por ejemplo el conocimiento del usuario sobre I8 ha sido incrementado en dos grados, pasando de “bajo” a “alto”.

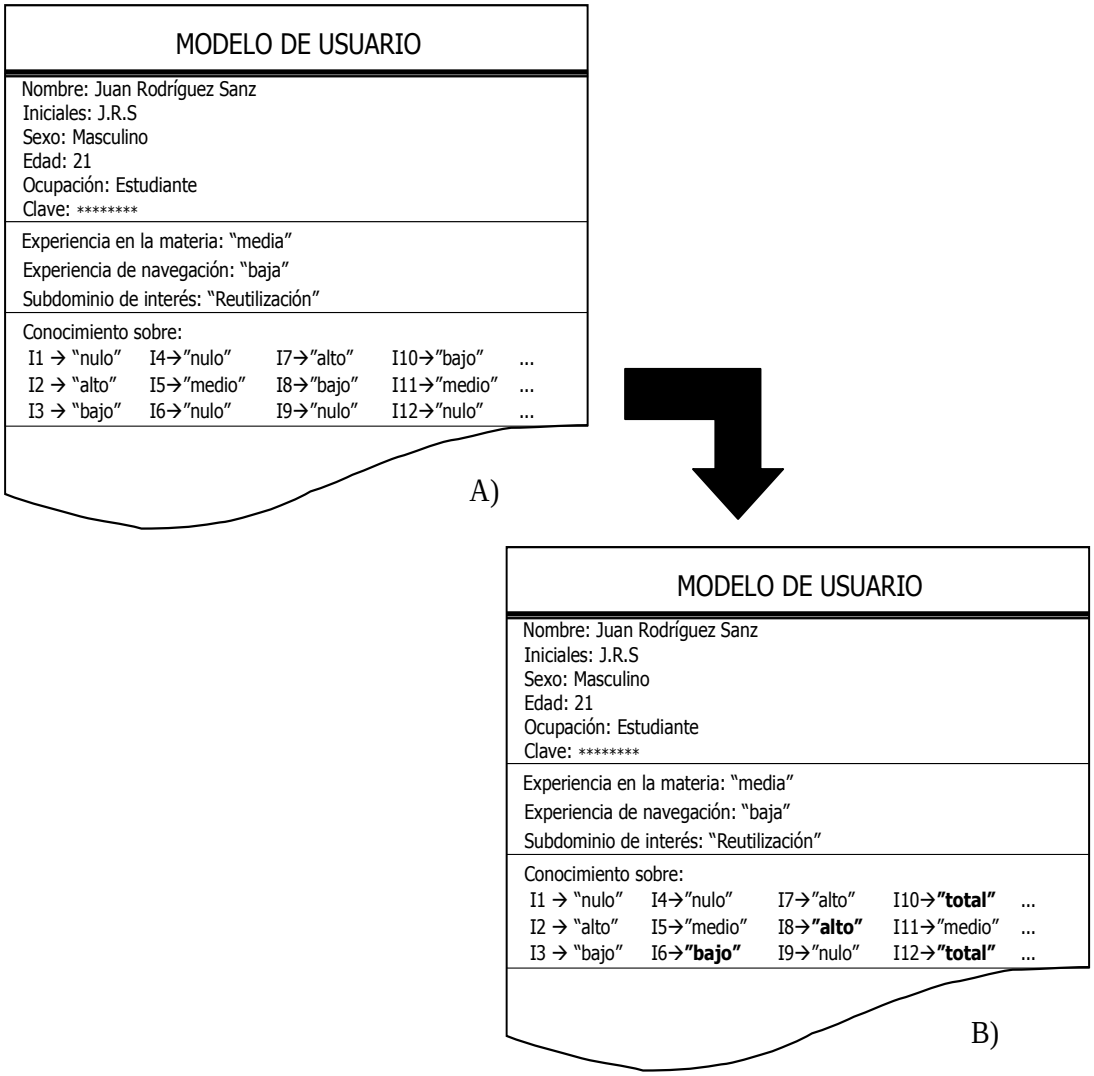


Figura 5: Actualización del estado de conocimiento

Aunque no se haya incluido en la figura, el modelo de usuario también guarda el número de visitas que se han realizado a cada ítem. Esta información es utilizada, por ejemplo, para validar que es la primera vez que Juan visita el ítem I12, ya que sólo en ese caso se debe actualizar I6.

De primeras es posible pensar que el conocimiento sobre un ítem nos indica si éste ha sido o no visitado anteriormente. Sin embargo, esto es no es siempre así. Un ítem puede tener conocimiento "nulo" y haber sido visitado por el usuario. Esta situación ocurre cuando el usuario tiene acceso al contenido del ítem, pero no cumple las restricciones de accesibilidad necesarias para que su regla de actualización sea ejecutada. Del mismo modo, un ítem puede tener conocimiento mayor que "nulo" sin haberse visitado nunca. En este caso el conocimiento adquirido sobre el ítem se habría obtenido con la visita a otro ítem, en cuya regla de actualización se incluye una actualización sobre éste.

3.3 Modos de navegación

La gran mayoría de los sistemas hipermedia y sistemas web en general, definen un único modo de navegar la información proporcionada. Los sistemas hipermedia

adaptativos incorporan técnicas de adaptación que consiguen hacer más cómoda y personal la navegación del usuario, pero siempre de acuerdo al modo estipulado. Nosotros proponemos un **esquema multi-modal de navegación**, conscientes de que, dependiendo de sus circunstancias y del fin que persigue, el usuario deseará recorrer la información de una forma u otra.

Por ejemplo, un usuario novato, con tiempo y disposición para aprender, preferirá seguramente una navegación guiada, en la que el sistema le obligue a visitar la información en un orden adecuado con los requisitos pedagógicos establecidos por el autor en las reglas de conocimiento.

Mientras que este mismo control puede resultar engorroso para un usuario más avanzado, que con relativa prisa, desea acceder a un ítem de información para consultar un dato concreto.

La figura 6 muestra los cuatro tipos de navegación permitidos: tradicional, por conceptos, por relación conceptual y por conocimiento. Tal y como se muestra en la figura los dos primeros, navegación tradicional y por conceptos, son modos de navegación libre (sección 3.3.1) donde no existen restricciones de acceso. Mientras que en los dos últimos, navegación por relación conceptual y por conocimiento, el acceso a determinados ítems está restringido (3.3.2), en el primer caso por restricciones de orden y en el segundo por restricciones de conocimiento. El grado de adaptación llevado a cabo en los distintos modos crece de acuerdo al color con el que aparecen representados en la figura, esto es, un color más oscuro indica un mayor grado de adaptación al usuario.

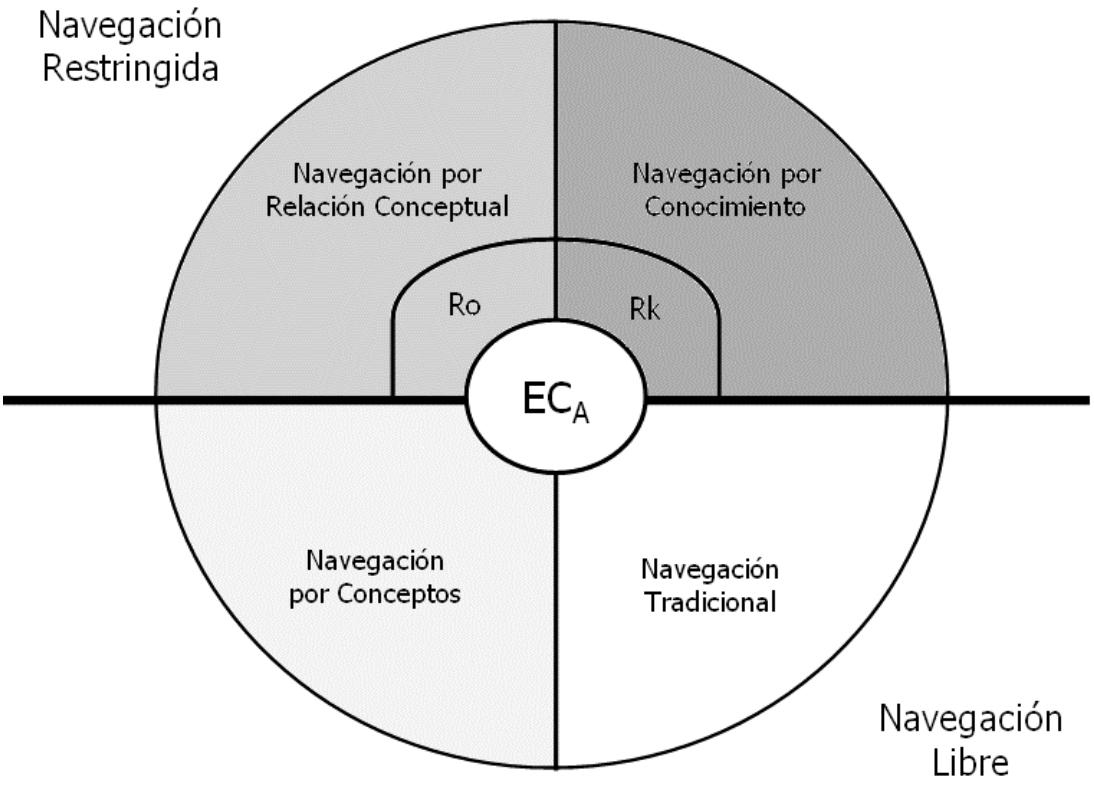


Figura 6: Modos de navegación

Es conveniente recordar que, en cualquiera de los cuatro modos de navegación permitidos, el sistema controla el proceso de adquisición de conocimiento del usuario en la forma descrita en la sección anterior.

3.3.1 Navegación Libre

La navegación libre, esto es, sin restricciones de acceso a la información, puede causar problemas de comprensión en el usuario si éste accede al contenido de un ítem cuya información es demasiado complicada para su estado de conocimiento. Sin embargo, la utilización de este tipo de navegación es una elección que libremente realiza el usuario. Ya que el sistema pone a su disposición dos tipos de navegación restringida (véase sección 3.3.2) que permiten paliar en gran parte este tipo de inconvenientes.

3.3.1.1 Navegación Tradicional

Este modo de recorrer la información es el más cercano al esquema de navegación al que los usuarios web están acostumbrados. Esto es, el usuario lee los trozos de información en el orden que le apetece, sin ningún tipo de guía, restricción o consejo al respecto.

La diferencia radica, no en el modo de recorrer la información, sino en la "plataforma" proporcionada para ello. Y es que, en nuestro caso, el usuario selecciona los ítems de información directamente sobre una red semántica que muestra las relaciones existentes entre los conceptos e ítems de la parcela de conocimiento que se explora. Esta red semántica es la representación de la EC_P^i incluida en la estructura de aprendizaje elegida, de forma personalizada, para el usuario.

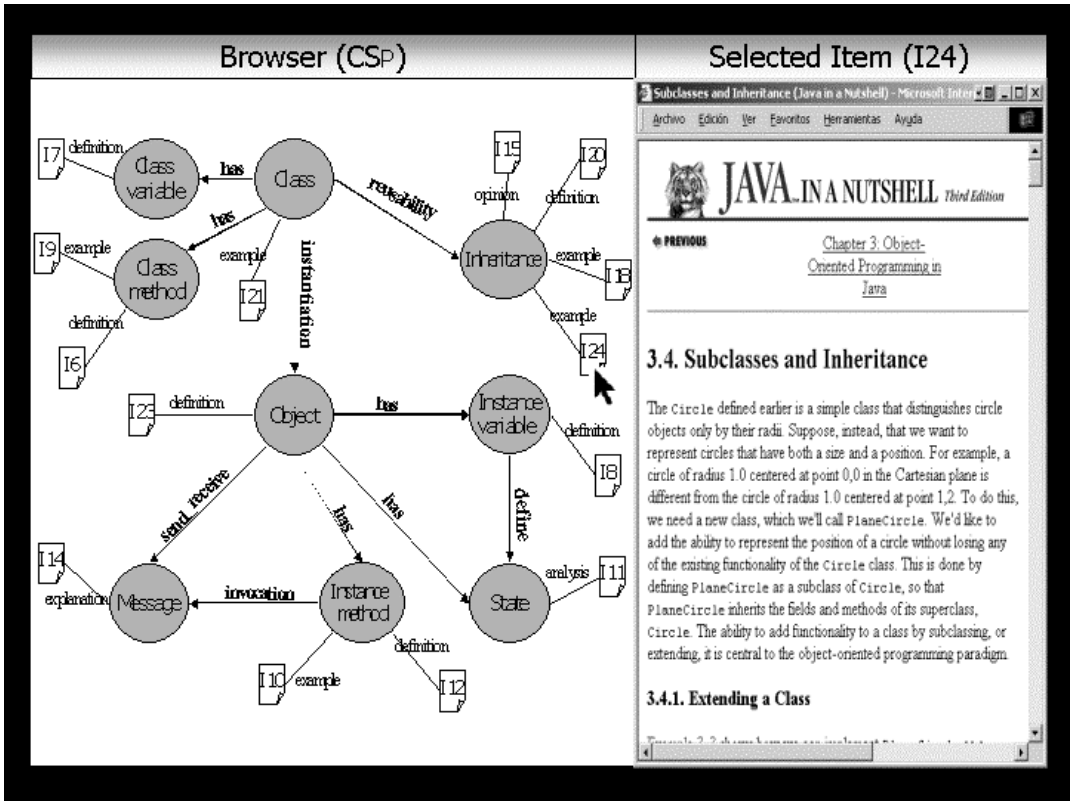


Figura 7: Navegación Tradicional

Como puede observarse en la figura 7, la interacción del usuario se realiza a través de una interfaz con dos paneles:

- el panel izquierdo proporciona la estructura de navegación, mientras que
- el panel derecho visualiza el contenido del ítem seleccionado (I24 en el ejemplo).

Navegar sobre la red semántica, tiene un claro beneficio para la orientación del usuario y es que, en todo momento conoce el ítem seleccionado, el concepto sobre el que trata, otros ítems asociados al mismo concepto y conceptos relacionados con el concepto actual. Aún así, si la presentación es extensa, un usuario novato puede desorientarse. Por lo que, este tipo de navegación se recomienda sólo a los usuarios con alto conocimiento en la materia, que tienen clara la información que buscan y desean ir directamente hacia ella sin ningún tipo de restricción que los demore.

3.3.1.2 Navegación por Conceptos

El modo de navegación explicado en la sección anterior, puede causar problemas de desorientación si la presentación incluye un número elevado de ítems de información. Además, el usuario puede estar interesado en visualizar y navegar el dominio de conocimiento del sistema hipermedia al completo. En este caso, la presentación proporcionada incluiría el conjunto íntegro de conceptos, ítems y asociaciones de la EC_M. Obviamente, en esta situación, la sensación de pérdida puede sobrevenir al usuario al contemplar tal volumen de información, aunque se presente etiquetada y organizada semánticamente.

Para disminuir los problemas de desorientación que pueden acontecer durante la navegación tradicional, se ofrece al usuario un modo de navegación libre basado en conceptos que reduce considerablemente el tamaño de la estructura de navegación ofrecida. Para conseguirlo, la "plataforma" de navegación es la red semántica que representa la parcela de conocimiento seleccionada, pero en esta ocasión, desprovista visualmente del dominio de información. Esto es, sin ítems ni asociaciones funcionales.

En la figura 8 se muestra la interfaz proporcionada a un usuario durante el modo de navegación por conceptos. La red semántica aparece situada en el panel superior izquierdo e incluye todos los conceptos y relaciones conceptuales de la EC_M de la figura 1. Sin embargo, al ocultar el dominio de información, presenta un tamaño mucho más reducido que ésta.

Este modo de navegación permite un nivel de abstracción superior, ya que la unidad de navegación no es el ítem sino el concepto en sí mismo. Al seleccionar un concepto, el usuario obtiene una serie de información relativa al mismo. Esta información no es otra que la contenida en los ítems que se asocian funcionalmente al concepto, eso sí, estructurada y organizada adecuadamente.

En el ejemplo de la figura 8 podemos ver cómo al hacer "clic" sobre el concepto *Inheritance*, el usuario obtiene:

En el panel inferior izquierdo, un listado de los ítems (I15, I20,...) ligados al concepto seleccionado, indicando nombre y propiedades (rol, autor, fecha de creación, dificultad, etc.).

a) En el panel derecho, un documento html que muestra diversa información sobre el concepto seleccionado. Ese documento es, en cierto modo, un resumen del concepto formado por todos o algunos de los ítems ligados al mismo.

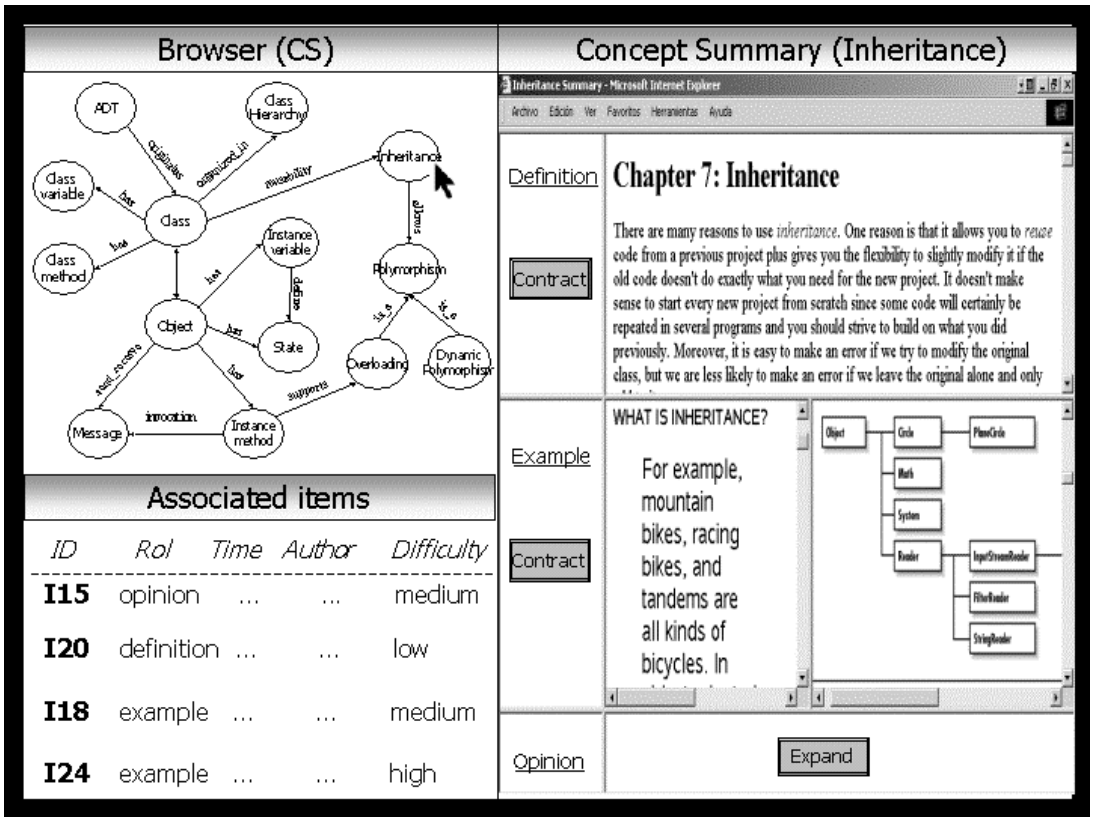


Figura 8: Navegación por Conceptos

Como vemos en el resumen generado para el concepto *Inheritance*, la información del documento se estructura en distintos marcos o secciones. La forma en que se organiza esta información es inicialmente determinada por el autor, pero después puede ser modificada por el usuario de acuerdo a sus preferencias, pasando a formar parte de su modelo de usuario. De este modo, la estructura de los resúmenes se adapta a los gustos particulares de cada usuario.

En el ejemplo, se pueden observar que las tres primeras secciones del resumen son: 1º definiciones, 2º ejemplos y 3º opiniones.

Dentro de cada sección aparece el contenido de los ítems asociados al concepto mediante el rol correspondiente a esa sección.

En el ejemplo, la primera sección visualiza el contenido del ítem I20 ligado con el rol "definition" al concepto *Inheritance*. Del mismo modo, la segunda sección muestra dos ejemplos de herencia contenidos en los ítems I18 e I24 respectivamente.

La última sección, dedicada al rol "opinión" mostrará el contenido del ítem I15 sólo si el usuario selecciona el botón <<*Expand*>> incluido en dicha sección.

Esto se debe a que, inicialmente el autor y después el usuario pulsando el botón <<*Contract*>>, puede establecer como opcionales aquellos roles que considere accesorios o poco relevantes para comprender el concepto. Las secciones de estos roles aparecen contraídas lo que permite reducir la longitud del resumen.

3.3.2 Navegación Restringida

Durante la navegación restringida se aplican técnicas de adaptación que permiten personalizar la estructura de enlaces con la intención de dirigir al usuario durante su proceso de navegación, impidiéndole el acceso a determinada información y resaltando la conveniencia de otra, siempre, claro está, de acuerdo a las condiciones establecidas por el autor.

Según el carácter de estas condiciones distinguimos dos formas de navegación restringida. La primera, navegación por relación conceptual (sección 3.3.2.1) tiene en cuenta la historia de navegación del usuario en cuanto a qué ítems ha visitado con anterioridad. Mientras que la segunda, navegación por conocimiento (sección 3.3.2.2) encauza la navegación del usuario teniendo en cuenta, no los ítems visitados, sino el estado de conocimiento alcanzado.

3.3.2.1 Navegación por Relación Conceptual

Este tipo de navegación se basa en las restricciones de navegabilidad y las reglas de orden establecidas en el Sistema de Navegación del modelo SEM-HP. La idea es que el autor fije el sentido en que cada relación conceptual puede ser navegada, esto es, sólo desde el origen al destino o también desde el destino al origen. Una vez establecida la navegabilidad, las relaciones conceptuales se transforman en enlaces de navegación entre los conceptos implicados.

Así, en la EC_P de la figura 2, si asumimos que todas las relaciones existentes son navegables únicamente en el sentido de la flecha, obtendríamos, entre otras, las siguientes restricciones de navegación que vienen dadas por el propio grafo:

Tabla 8: Restricciones de navegación

a)	Estando en el concepto *Object*, es decir, siendo el ítem I23 el último ítem visitado, es posible acceder a cualquier ítem de los conceptos: *Instance variable* (I8), *State* (I11), *Instance method* (I10 e I12) y *Message*(I14).
b)	Para acceder al ítem I14, asociado al concepto *Message*, es necesario haber visitado previamente algún ítem asociado al concepto *Instance method* (I10 o I12) o al concepto *Object* (I23).

Si nos centramos en la restricción b) vemos que son posibles dos caminos para llegar a I14, en el primero se sigue la relación conceptual "invocation" y en el segundo se llega a través de la relación "send_receive". Esto implica que necesariamente el ítem visitado en el momento previo a la selección de I14 debe estar asociado al concepto origen de alguna de estas dos relaciones: *Message* e *Instance method* respectivamente.

Además de estas restricciones de orden derivadas directamente de la navegabilidad de las relaciones conceptuales, el autor puede añadir nuevas restricciones para la visita a un ítem. Estas restricciones adicionales exigen o prohíben la visita anterior, en dos o más pasos de navegación, a un ítem I_k para poder acceder a otro ítem I_j. Las restricciones pueden ser diferentes para cada uno de los caminos posibles hasta I_j. Por ejemplo, el autor puede modificar la restricción b) de la tabla 8, dando lugar a las siguientes reglas de orden:

Tabla 9: Reglas de orden para I14

b.1)	Para acceder al ítem I14, asociado al concepto *Message*, es necesario haber visitado previamente algún ítem asociado al concepto *Instance method* (I10 o I12) y haber visitado con anterioridad el ítem I11.
b.2)	Para acceder al ítem I14, asociado al concepto *Message*, es necesario haber visitado previamente el ítem I23 asociado al concepto *Object* y haber visitado con anterioridad el ítem I10 y el ítem I12.

Para que el usuario pueda acceder a un ítem basta con que satisfaga todas las restricciones en uno de los caminos que conducen hasta él.

Por ejemplo, de acuerdo a la regla de orden b.2) el usuario puede visitar el ítem I14 si el último ítem que ha visitado es I23 (la definición de *Object*) y en algún momento anterior visitó los ítems I10 e I12 que contienen información acerca del concepto relacionado *Instance method*.

Así, para saber qué ítems puede visitar el usuario y cuáles no, el sistema necesita conocer, no sólo el último ítem visitado, sino todos los ítems visitados en el pasado. Esto confirma la necesidad de almacenar las visitas realizadas por el usuario sobre cada ítem en su modelo de usuario.

El Sistema de Aprendizaje se encarga de personalizar la estructura de navegación proporcionada al usuario en el modo de navegación por relación conceptual. La finalidad principal de la adaptación realizada puede desglosarse en dos objetivos:

Por un lado, evitar que el usuario acceda al contenido de un ítem prohibido.

Y por otro, recordar al usuario su historia de navegación.

El primer objetivo se consigue ocultando y deshabilitando los ítems cuya visita tiene prohibida el usuario. La *deshabilitación* le impide acceder al contenido del ítem y la *ocultación* le disuade de intentarlo. El segundo objetivo se resuelve *adjuntando junto a cada ítem el número de visitas realizadas*. Esta información permite evitar visitas repetidas no deseadas y hace más consciente al usuario de qué puede hacer para que determinados ítems prohibidos se vuelvan accesibles (sobre todo si las reglas de autor se hacen públicas).

En el ejemplo de la figura 9 se muestra la adaptación realizada para un usuario, cuya historia de visitas se anota sobre la propia estructura de navegación.

Teniendo en cuenta que el último ítem visitado es I23, véase como se resalta el **concepto actual** (*Object*) y los **conceptos alcanzables** desde éste siguiendo una relación conceptual (*Message*, *Instance method*, *State* y *Instance variable*).

Nótese, también, que el ítem I14, aunque asociado a un concepto resaltado, aparece oculto debido a que el usuario no satisface la restricción (haber visitado antes I10) que el autor ha añadido para su visita en la regla de orden b.2 de la tabla 9.

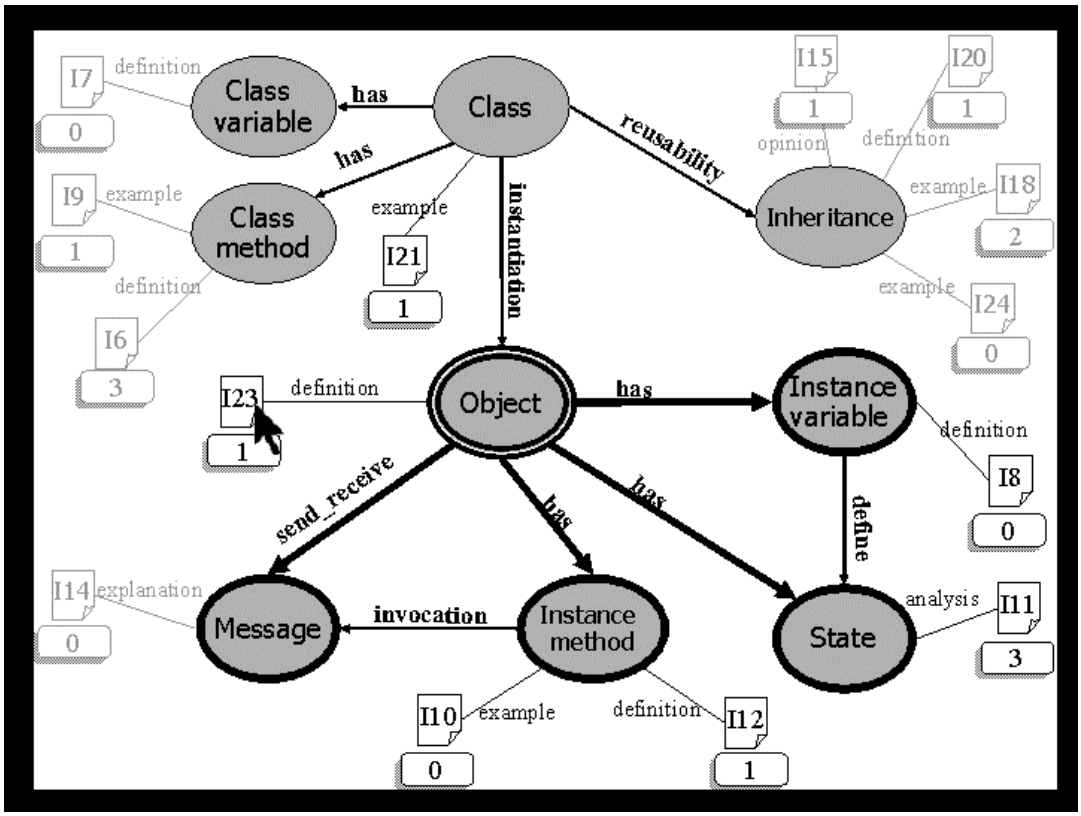


Figura 9: Adaptación durante Navegación por Relación Conceptual

3.3.2.2 Navegación por Conocimiento

En este tipo de navegación, las reglas de conocimiento no se utilizan únicamente para determinar si una actualización debe ser o no ejecutada tras la visita de un ítem, sino que, de su evaluación se deduce si un ítem puede ser visitado y si esta visita es relevante. Por lo tanto, la navegación del usuario es guiada en función de los prerrequisitos pedagógicos expresados por el autor en las reglas de conocimiento.

No se trata de qué ítems debe haber visitado antes el usuario, sino de qué ítems debe conocer y con qué nivel. Es decir, si es necesario poseer un nivel de conocimiento "alto"sobre I10 e I12 para acceder a I9, no significa que obligatoriamente haya que visitar I10 e I12 antes que I9. Puesto que el usuario puede conseguir conocimiento sobre estos ítems visitando otros ítems relacionados (véase la sección 2.2).

Puesto que el usuario sólo puede visitar un ítem si satisface sus restricciones de accesibilidad, en este tipo de navegación, siempre la visita de un ítem va a llevar asociada la ejecución de su regla de actualización.

La navegación por conocimiento es el modo más apropiado para los lectores inexpertos que quieran aprender, de una manera exhaustiva y correcta, un determinado subdominio de conocimiento. Por otro lado, es el más cercano al modelo educacional usual, en el cuál el tutor dirige el aprendizaje del estudiante, presentándole la información en un orden didáctico y orientado a su nivel de conocimiento específico.

3.3.2.2.1 Adaptación de ítems accesibles e idóneos

Al igual que en la navegación por relación conceptual, en la estructura de navegación se ocultan y deshabilitan los ítems prohibidos. En este caso, un **ítem** es prohibido o **inaccesible** cuando el estado de conocimiento del usuario no satisface las restricciones de accesibilidad impuestas por el autor en ninguna de las reglas de conocimiento asociadas a dicho ítem.

Por otro lado, los **ítems accesibles pero no idóneos** son deslucidos visualmente, desaconsejando al usuario que acceda a su contenido, aunque esta visita le esté permitida. Finalmente, un **ítem** es **accesible e idóneo** sólo si el usuario satisface completamente alguna de sus reglas de conocimiento, es decir, el conjunto total de restricciones de accesibilidad e idoneidad impuestas en la regla.

La siguiente figura muestra la adaptación realizada para el usuario cuyo estado de conocimiento es el que se muestra, de forma incompleta, en el modelo de usuario de la figura 5.A.

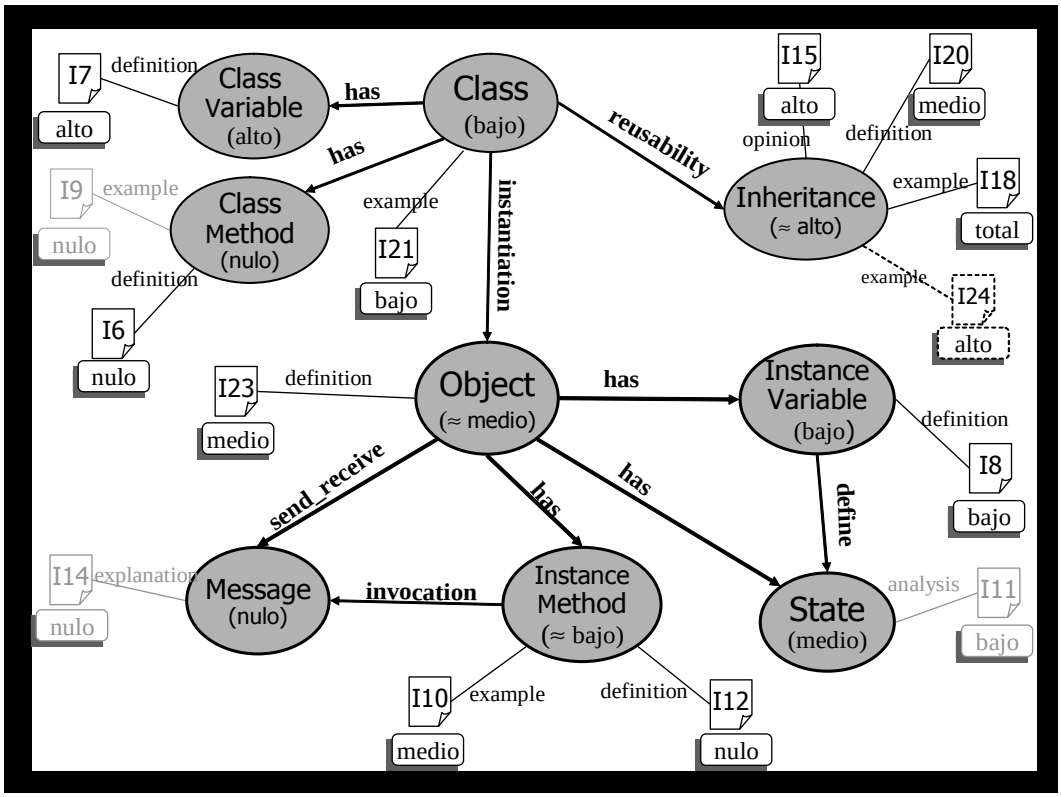


Figura 10: Adaptación durante Navegación por Conocimiento

Como puede observarse, *el conocimiento del usuario es anotado sobre la propia estructura de navegación* utilizando para ello etiquetas semánticas, perfectamente inteligibles para él. Esta anotación hace partícipe al usuario de su proceso de

aprendizaje, ya que puede ver cómo aumenta su conocimiento conforme inspecciona los ítems de la presentación.

Además, dentro de cada concepto se indica el grado con que el usuario lo conoce. Este conocimiento es obtenido mediante la aplicación de las reglas de peso.

Véase, por ejemplo, que el conocimiento anotado para el concepto *Inheritance* (≈ "alto") se corresponde con el resultado de ejecutar la regla de peso definida para dicho concepto en la tabla 6, utilizando el conocimiento que el usuario posee acerca de los ítems asociados: I15, I20, I18 e I24.

Para realizar las operaciones indicadas en la regla peso, es necesario asignar un valor numérico a cada etiqueta semántica y una vez obtenido el resultado hacer la transformación inversa. En ocasiones el resultado numérico que se obtiene se encuentra entre dos etiquetas semánticas, haciéndose necesaria una aproximación a la más cercana. En esos casos se utiliza para la anotación el símbolo ≈ en lugar de =.

La regla de peso de un concepto implica a todos los ítems asociados a él, estén o no presentes en el subdominio de conocimiento que actualmente se navega. Por este motivo, si el usuario observa que, habiendo logrado conocimiento "total" sobre todos los ítems ligados al concepto c_k en la estructura actual, no conoce totalmente dicho concepto, debe saber que tiene que recorrer otras estructuras si quiere alcanzar el grado de conocimiento "total" sobre c_k.

En la figura 10, algunos ítems de la estructura aparecen ocultos. Concretamente si nos fijamos en I9, vemos que esto ocurre porque el usuario no satisface ninguna de las reglas de conocimiento impuestas para su visita en la tabla 1.

El usuario puede seleccionar cualquier ítem, pero si selecciona un ítem oculto no obtendrá el contenido del mismo ya que la funcionalidad del enlace se ha eliminado. Suponiendo que el usuario visite I12 e I8, su nuevo estado de conocimiento, reflejado en el modelo de usuario de la figura 5.B, satisface las condiciones impuestas para acceder a I9 en la primera regla de conocimiento de la tabla 1. Por lo tanto, dicho ítem dejaría de estar oculto.

Adviértase también, que el ítem I24 aparece bordeado con línea discontinua. Esto permite informar al usuario de que se trata de un ítem demasiado simple para él y por lo tanto, no idóneo. Esto ocurre porque el usuario no satisface la restricción de idoneidad impuesta para I24 en su única regla de conocimiento (tabla 2). Concretamente, según el autor, si se tiene un conocimiento "total" sobre I18 la lectura de I24 no proporciona conocimiento nuevo. Sin embargo, el usuario es libre de visitar I24, aunque esta visita esté desaconsejada, ahora y en adelante.

3.3.2.2.2 Adaptación a la meta

El usuario puede especificar, si lo desea, una **meta de conocimiento**. La meta es un estado de conocimiento que el usuario desea alcanzar. Para establecer su meta, el usuario sólo debe especificar el grado de conocimiento que quiere lograr sobre aquellos ítems o conceptos que le interesen especialmente. Esto es, la meta no tiene por qué incluir a todos los ítems o conceptos existentes, sino sólo a aquellos que llaman la

atención del usuario. Por ejemplo, una meta sobre la estructura de navegación de la figura 10, podría ser la que se recoge en la tabla 10.

Tabla 10: Una meta de conocimiento

Deseo alcanzar un conocimiento "total" sobre el ítem I8, un conocimiento "alto" sobre I9 y un conocimiento "medio" sobre I21.

Un estado de conocimiento satisface una meta si presenta para cada ítem o concepto incluido en ésta un grado de conocimiento igual o superior al especificado. Por lo tanto, normalmente van a existir múltiples estados de conocimiento que satisfagan la meta del usuario.

La meta del usuario forma parte de su modelo de usuario y es utilizada por el sistema para asistir al usuario en la consecución de sus objetivos. Este soporte a la meta es realizado a dos niveles distintos:

Consejo local: Indicar al usuario en cada paso que ítems puede visitar para acercarse a su meta.

Consejo global: Proporcionar al usuario una secuencia de ítems que debe visitar en un orden determinado para lograr la meta.

El **consejo local** se realiza resaltando positivamente sobre la estructura de navegación un conjunto de ítems deseables. Un **ítem** es **deseable** si su visita contribuye a alcanzar la meta impuesta sobre un determinado ítem o concepto. Obviamente, un ítem meta es deseable porque al visitarlo el usuario gana conocimiento sobre él. Pero, ¿qué pasa si un ítem meta no es accesible para el usuario?. En este caso, el ítem no se proporciona como deseable ya que no puede ser visitado por el usuario. En su lugar, el sistema considera deseable la visita de aquellos ítems para los que existe una restricción de accesibilidad no satisfecha en alguna regla de conocimiento definida para el ítem meta. La visita de estos ítems deseables tiene el objetivo de que el ítem meta se vuelva accesible. De nuevo, si alguno de estos ítems no fuese accesible el sistema propone como deseables el conjunto de ítems requeridos para que así sea.

Dada la meta expresada en la tabla 10, las reglas de conocimiento de la tabla 1 y el estado de conocimiento anotado en la figura 10, el conjunto de ítems deseables se calcula de la forma que sigue:

Los ítems meta I8 e I21 son deseables puesto que son accesibles.

1. El ítem meta I9 no es accesible, por lo tanto, se consideran como deseables los ítems I6, I21, I12 e I10 cuya restricción de accesibilidad para visitar I9 no se satisface en el momento actual.

Observe que el ítem I23 también es requerido en la regla de conocimiento de I9, pero no se incluye como deseable porque su restricción ya está satisfecha.

Esta técnica permite resaltar en cada paso un conjunto de ítems cuya visita logra que el usuario se aproxime a la satisfacción de su meta de conocimiento. El usuario es libre de seleccionar uno de estos ítems u otro diferente. En cualquier caso, tras cada visita, es necesario recalcular el conjunto de ítems deseables para el siguiente paso.

La técnica de **consejo global** construye y proporciona una **ruta de navegación** capaz de conducir al usuario desde su estado de conocimiento actual hasta un estado de conocimiento meta. Para ello, el usuario debe visitar los ítems de la ruta en el orden que se le especifica.

El sistema obtiene la ruta realizando un proceso de búsqueda sobre el espacio de estados de conocimiento posibles. El nodo inicial de esta búsqueda es el estado de conocimiento actual del usuario y el nodo final es algún estado de conocimiento que satisface la meta impuesta por éste. Para optimizar esta búsqueda se aplican diversas estrategias: búsqueda heurística [Russell, 96], algoritmos Greedy [Brassard, 96] [Cornen, 97], etc.

Para que la ruta proporcionada se ajuste perfectamente a cada usuario, el sistema tiene en cuenta las preferencias del usuario respecto al contenido de cada ítem y a la longitud de la ruta. De esta forma dos usuarios con el mismo estado inicial y objetivo, obtendrán dos rutas diferentes en función de sus preferencias.

Las **preferencias del usuario** también forman parte de su modelo de usuario y permiten indicar las cosas que gustan o disgustan al usuario. Así para cada característica de un ítem: idioma, medio, rol, autor, fecha y dificultad, el usuario establece las cosas que le (dis)gustan ordenándolas de más a menos. También puede determinar la importancia que para él tiene cada atributo asignándole un peso mayor cuanto más relevante lo considere. Por ejemplo, el usuario puede indicar las preferencias que se muestran en la tabla 11.

Tabla 11: Preferencias del usuario

Respecto al idioma de un ítem, me gustan los ítems cuyo contenido está en: 1º. Español, 2º. Inglés y 3º. Francés.
Respecto al medio de un ítem: 1º.Me gustan las imágenes, 2º. No me gusta el audio y 3º. Me gusta el video.
.....
La distribución de pesos para las características es: 0,3 para “idioma”, 0,2 para “rol”, 0,3 para “medio”, 0,1 para “dificultad”, 0,1 para “fecha” y 0 para “autor”.

El usuario puede expresar sus preferencias con total flexibilidad, indicando cosas como que le gustan más las imágenes que el video y que le disgusta más el audio de lo que le gusta el video. O que, para él, las características más importantes de un ítem son el idioma y el medio que se ha utilizado para expresar la información, resultándole indiferente quién ha creado el ítem.

Además de sus gustos respecto al contenido del ítem, el usuario puede indicar su grado de interés en que la ruta sea corta. Si la preferencia por ruta corta es absoluta, el resto de preferencias se ignoran y el sistema realiza una búsqueda en anchura para obtener la ruta con menor número de visitas.

Si por el contrario, la preferencia por ruta corta es nula, el sistema realiza la búsqueda preocupándose fundamentalmente de que los ítems visitados en el camino tengan las características que más gustan al usuario. El sistema calcula para cada ítem candidato para formar parte de la ruta, un indicador que refleja en qué grado se acerca o aleja de los gustos del usuario. Para obtener este indicador se tiene en cuenta el valor del ítem para cada característica (si gusta o disgusta y cuánto) y la importancia atribuida a ésta.

Lo normal es que ambas cosas preocupen al usuario, la longitud de la ruta y las características de los ítems que la componen, por ello durante la búsqueda se combinan ambas preferencias en el grado indicado.

4. INTEGRACIÓN DEL SISTEMA DE APRENDIZAJE EN SEM-HP

El Sistema de Aprendizaje descrito conceptualmente a lo largo de este capítulo constituye en sí mismo un modelo de adaptación que podría incorporarse a distintos sistemas o arquitecturas hipermedia, de forma más o menos inmediata. Claro está, siempre que éstos satisfagan unos requisitos básicos, entre los que cabría destacar la representación del dominio de conocimiento mediante redes semánticas o mapas conceptuales.

Sin embargo, el Sistema de Aprendizaje ha sido concebido y especialmente diseñado para ser integrado en el modelo SEM-HP, dotando a éste de la capacidad de adaptación al usuario y beneficiándose a cambio del robusto soporte que dicho modelo proporciona en cuanto a representación y evolución del conocimiento del sistema hipermedia.

Esta integración obliga a resolver dos aspectos importantes. Por un lado de qué manera el Sistema de Aprendizaje se ve afectado por los cambios sucedidos en el resto de sistemas de SEM-HP. Y por otro lado, cómo el conocimiento que el Sistema de Aprendizaje es capaz de extraer a partir de la navegación de los usuarios puede mejorar al resto de los sistemas. El primer aspecto es resuelto mediante la propagación del cambio, sección 4.1, mientras que el segundo requiere un nuevo mecanismo de adaptación explicado en la sección 4.2.

4.1 Propagación del cambio

Los distintos elementos que componen el Sistema de Aprendizaje son definidos sobre las estructuras conceptuales desarrolladas previamente para el resto de sistemas que constituyen SEM-HP. Concretamente la estructura de las reglas de peso y de una parte importante del modelo de usuario depende directamente de los ítems, conceptos y asociaciones existentes en la EC_M del Sistema de Memorización. Del mismo modo que las reglas de actualización y de conocimiento se definen para los ítems incluidos en una EC_P creada y almacenada en el Sistema de Presentación.

La interrelación que existe entre el Sistema de Aprendizaje y el resto de sistemas, requiere que cuando se modifique un elemento en alguno de estos sistemas sea necesario evaluar el alcance del cambio. Y, en caso de ser preciso, propagarlo, realizando las modificaciones pertinentes en los elementos del Sistema de Aprendizaje que se hayan visto afectados. Todo ello con el objetivo de que el conjunto de sistemas sea consistente desde una perspectiva global.

Este mecanismo de propagación no debe dejarse en manos del autor ya que de ser así, se complicaría bastante el proceso de evolución, cargando al autor con la responsabilidad de garantizar la integridad del sistema, y permitiendo que éste quedase en un estado inconsistente debido a un error, descuido o intención del autor.

Cuando el autor realiza un cambio, siempre a través de la acción evolutiva correspondiente, el sistema inicia de forma automática un proceso de propagación, desencadenando la ejecución de las acciones evolutivas del Sistema de Aprendizaje que sean necesarias. En la siguiente tabla se resume, muy brevemente, la **propagación del cambio** realizada sobre el Sistema de Aprendizaje. Indicando para cada uno de sus elementos los cambios producidos en el resto de sistemas que pueden requerir la ejecución de una acción evolutiva.

Tabla 12: Propagación sobre el Sistema de Aprendizaje

Elemento	Cambio que puede afectarle
Modelo de usuario	Crear o borrar un ítem o un concepto en la EC_M.
Reglas de peso	Crear o borrar un concepto o una asociación funcional en la EC_M.
Reglas de actualización	Mostrar u ocultar un ítem en la EC_P.
Reglas de conocimiento	Ocultar un ítem en la EC_P.

4.2 Adaptación por retroalimentación

El Sistema de Aprendizaje mantiene una representación interna del usuario con información muy completa acerca de éste. Una característica interesante del Sistema de Aprendizaje es su capacidad de aprovechar este conocimiento adquirido sobre el usuario para "perfeccionar" las estructuras definidas en el resto de sistemas que componen la arquitectura SEM-HP, especialmente en el Sistema de Memorización y de Presentación.

Sin embargo, no tendría mucho sentido redefinir las estructuras para ajustarlas a un usuario concreto. Esto daría lugar a estructuras individuales, cuando lo que se desea es todo lo contrario: estructuras genéricas que puedan ser navegadas por usuarios muy distintos, y luego, adaptadas a las diferencias de cada uno de ellos. Por lo tanto, las modificaciones propuestas deben atender a las necesidades de un **grupo de usuarios** y no de un usuario concreto.

Para que el análisis sea realmente significativo sólo se estudia el comportamiento de aquellos usuarios que navegan libremente, esto es, sin obligación de seguir los caminos pre-trazados por el autor. Durante el análisis, el sistema superpone los recorridos realizados por los distintos usuarios, con el objetivo de averiguar las estrategias de navegación utilizadas colectivamente por aquellos que navegan libremente.

Durante este proceso se construyen matrices de transición que permiten reflejar de forma clara la aceptación o el rechazo de los usuarios por cada asociación conceptual posible. Una matriz de transiciones tiene una fila y una columna por cada concepto incluido en la estructura conceptual analizada.

En la figura 11 se muestra parcialmente la matriz de transiciones mantenida en un determinado momento para la EC_M de la figura 1. El valor de la celda marcada con un círculo indica el número de veces que un usuario ha pasado de visitar un ítem asociado al concepto *Class* a visitar un ítem del concepto *Class method*. El hecho de que este número sea relativamente elevado nos indica que dicha relación, existente en la EC_M, es aceptada mayoritariamente por sus usuarios.

Por el contrario, el valor alto de la celda marcada con un triángulo nos avisa de que la mayoría de los usuarios aceptan una relación conceptual desde *Instance method* a *Class method*. Y, sin embargo, esta relación no existe en la EC_M definida por el autor.

	ADT	Class variable	Class method	Message	Instance method	...
ADT		12	7	5	23	
Class variable	25		134	3	34	
Class method	30	102		43	125	
Message	15	35	40		67	
Instance method	20	36	106	99		
Class	89	100	98	34	32	
...						

Figura 11: Matriz de transiciones

La comparación de las **matrices de transición** con las estructuras reales definidas por el autor permite detectar asociaciones conceptuales aceptadas por la mayoría de los lectores que no han sido consideradas por el autor. Así como conceptos, ítems y asociaciones conceptuales incluidas en las estructuras existentes pero que raramente son utilizadas por sus usuarios.

Tras el análisis realizado, el sistema sugiere distintas modificaciones al autor, quién realizando los cambios propuestos puede ajustar sus representaciones al modelo mental que de ellas tienen los usuarios del sistema. Por ejemplo, ocultando/mostrando relaciones en una presentación, o borrando/creando asociaciones en la estructura de memorización. Esta técnica también permite detectar conceptos, ítems y presentaciones que los usuarios ignoran, así como el perfil de las presentaciones más/menos utilizadas.

Denominamos **realimentación adaptativa** a este tipo de adaptación porque consigue que la estructura del sistema hipermedia se redefina en función de la navegación libre realizada sobre la misma por un grupo significativo de usuarios.

5. ESTRUCTURA DE LA FORMALIZACIÓN

Durante la exposición realizada en las secciones anteriores han ido apareciendo, de modo informal, los principales elementos que constituyen el Sistema de Aprendizaje propuesto y desarrollado en la presente tesis. En los posteriores capítulos, cada uno de estos elementos es definido, descrito y formalizado convenientemente. Esta aportación se divide en dos importantes módulos, en función del carácter estructural o funcional del elemento considerado. Concretamente, el primero de estos módulos recoge los **elementos estructurales** del Sistema de Aprendizaje, mientras que el segundo está dedicado a los **elementos funcionales**.

Obviamente, ambos módulos están estrechamente relacionados puesto que los elementos estructurales posibilitan y determinan el funcionamiento del sistema. Dicho funcionamiento depende, pues, de las estructuras que lo soportan. Pero además, permite detectar posibles mejoras sobre estas estructuras. Indiscutiblemente, estas modificaciones después repercutirán sobre el funcionamiento del sistema.

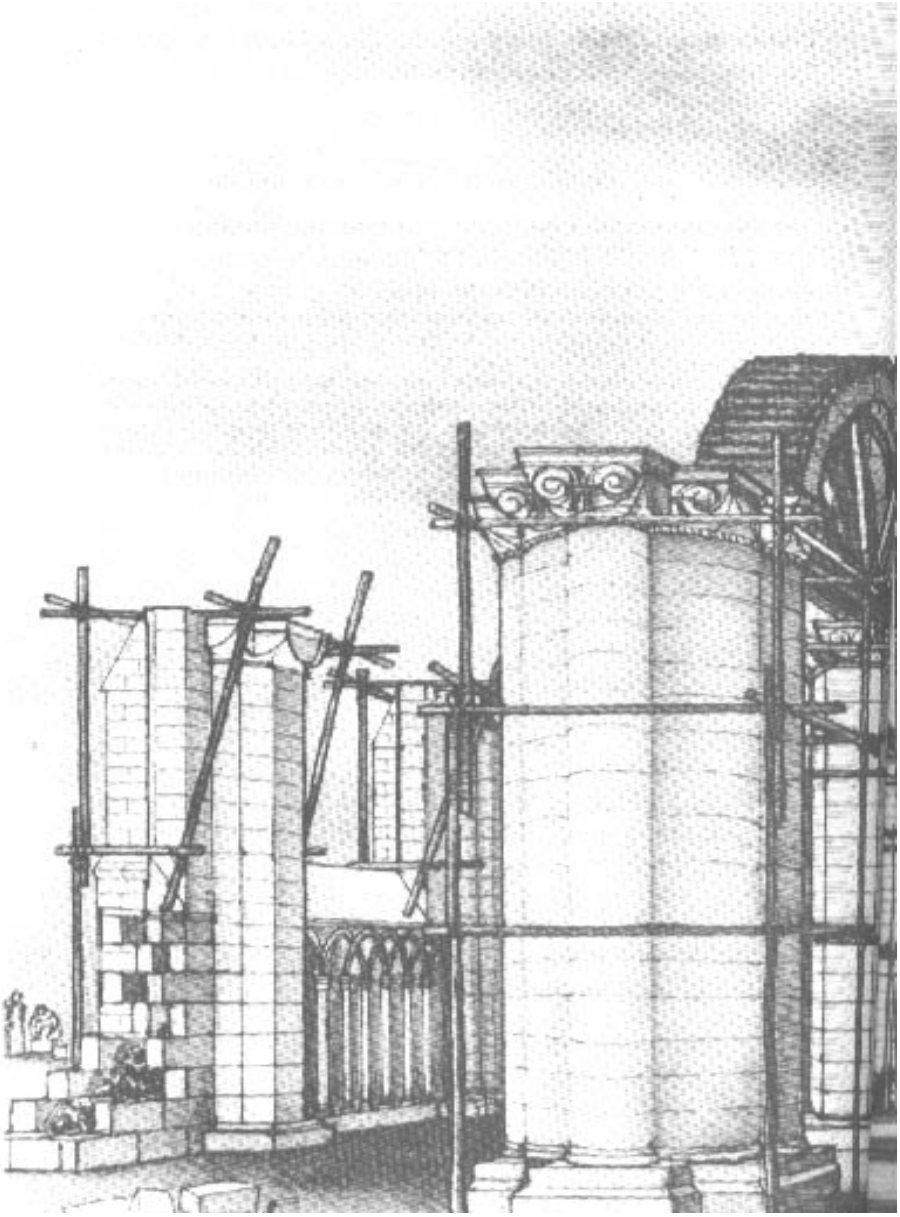
En la siguiente tabla se indica el módulo al que pertenece cada elemento, indicando además el capítulo concreto en el que se encuentra detallado.

Tabla 13: Estructura de la aportación

	Módulo	Elemento	Capítulo
Sistema de Aprendizaje	Elementos Estructurales (III)	Reglas de conocimiento	8
		Reglas de actualización	9
		Consistencia Rk ∪ Ru	10
		Reglas de peso	11
		Modelo de usuario	12
		Propagación externa del cambio	13
	Elementos Funcionales (IV)	Elección de la EC_A	14
		Modos de navegación	15
		Navegación por conceptos	16
		Navegación por relación conceptual	17
		Navegación por conocimiento	18
		Ítems deseables	19
		Rutas guiadas	20
		Retroalimentación adaptativa	21

Módulo III

Sistema de Aprendizaje: Elementos Estructurales

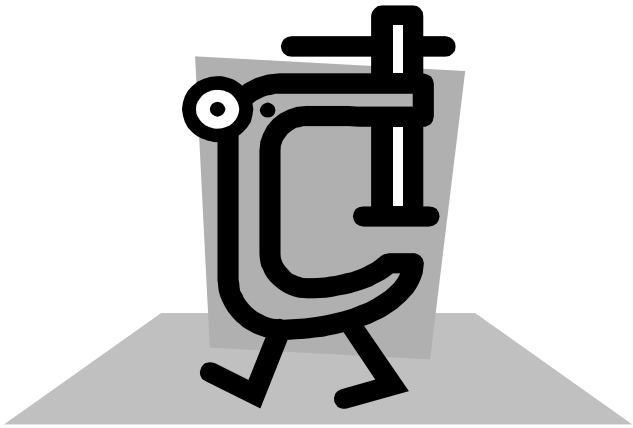


Los Pilares de la Tierra

Módulo III

CAPÍTULO 8

Reglas de Conocimiento



Resumen

Las reglas de conocimiento constituyen la herramienta lógica que se proporciona al autor para expresar prerrequisitos de conocimiento entre los ítems de información del sistema hipermedia. En este capítulo se definen los principales conceptos relativos al estado de conocimiento del usuario y a los requisitos de conocimiento establecidos por el autor. Se especifican formalmente dos lenguajes que permiten representar a nivel de autor e internamente las reglas de conocimiento. Se demuestra la completitud, consistencia e equivalencia de ambos lenguajes, justificando la necesidad y utilidad de una doble representación. Finalmente se establece un conjunto de acciones evolutivas que permite hacer evolucionar el conjunto de reglas de conocimiento en una forma flexible y consistente.

Tabla de contenidos

Reglas de Conocimiento

1. INTRODUCCIÓN

Las reglas de conocimiento dirigen la navegación del usuario en función de su propio conocimiento. De esta forma, un usuario podrá visitar unos ítems, y sin embargo, tendrá prohibido el acceso a otros dependiendo del grado de conocimiento que posea sobre determinados ítems. Además también informa de cuáles dejan de ser relevantes cuando el usuario supera un determinado estado de conocimiento. De esta forma, el usuario es consciente en todo momento de la información de cuya lectura puede prescindir, agilizándose así su proceso de adquisición de conocimiento.

Este mecanismo tiene como principal objetivo evitar que el usuario acceda a un ítem si no está preparado para comprender la información contenida en éste. Diremos que un usuario está preparado para asimilar el contenido de un ítem si posee un conocimiento previo suficiente sobre otros ítems que, para el autor, son **prerrequisitos pedagógicos** del ítem actual.

> **Def 8.1 [Regla de conocimiento]** Una regla de conocimiento define las restricciones de conocimiento (prerrequisitos pedagógicos) que debe cumplir el usuario para poder visitar un determinado ítem y para que dicha visita sea aconsejable.

Cada regla de conocimiento se asocia a un ítem concreto y establece qué otros ítems debe conocer el usuario, y con qué grado de conocimiento, para poder acceder de forma idónea a éste. No obstante, el autor puede especificar distintas reglas de conocimiento para un mismo ítem, indicando en cada regla, mediante diversas restricciones de conocimiento, una forma diferente de asegurar:

a) que el usuario está capacitado para comprender el ítem, esto es, posee la información previa necesaria para ello, y
b) que el contenido del ítem no es ni redundante ni demasiado simple para su nivel de conocimiento.

Cabe resaltar que inicialmente el conjunto de reglas de conocimiento para una estructura conceptual de aprendizaje (EC_A)(Def 6.15) es vacío, de modo que la visita a cualquier ítem es aconsejada sin necesidad de disponer de ningún conocimiento previo. Es una tarea exclusiva del autor definir las reglas de conocimiento que considere pertinentes de acuerdo a su conocimiento del tema y a sus estrategias pedagógicas.

2. ESTADO DE CONOCIMIENTO

Antes de profundizar en la sintaxis y semántica de las reglas de conocimiento, es necesario definir la noción de estado de conocimiento. Es decir, qué entendemos por estado de conocimiento y cómo se constituye.

> **Def 8.2 [Estado de conocimiento]** Un estado de conocimiento es una composición de múltiples conocimientos. El estado de conocimiento al que aquí haremos referencia se limita únicamente al dominio del sistema hipermedia. Por lo tanto,

está formado por el conocimiento que el usuario posee acerca de cada trozo de información existente en el sistema hipermedia. Su estado de conocimiento varía cada vez que el usuario modifica (aumenta o disminuye) algún conocimiento incluido en dicho estado.

Def 8.3 [Grado de conocimiento] El conocimiento del usuario acerca de un ítem o concepto se mide y expresa utilizando un grado de conocimiento. Este grado de conocimiento se representa mediante una etiqueta semántica, a la que de forma abreviada notaremos $\mathbf{et_{SEM}}$. Dicha etiqueta refleja el nivel de conocimiento del usuario, es decir, cuánto conoce acerca de un determinado elemento, abstracto o concreto (véase la tabla 1).

Durante la presente memoria, consideramos y proponemos cinco etiquetas semánticas diferentes; aunque, en principio, cada autor puede utilizar las que desee[1]. La elección del número cinco no es arbitraria, ya que dicho número ha sido utilizado ampliamente en los sistemas de calificación tradicionales (deficiente, suspenso, aprobado, notable, sobresaliente). Esto se debe a que con cinco niveles es posible representar dos valores extremos, el punto medio entre dichos extremos y, además, un valor por encima y por debajo de ese punto medio. Lo cual supone un buen equilibrio entre precisión y semántica. Con más niveles es difícil mantener un criterio uniforme, mientras que con menos se pierde expresividad.

Concretamente las cinco etiquetas proporcionadas para expresar un grado de conocimiento son: “nulo”, “bajo”, “medio”, “alto”, y “total”. La etiqueta “nulo” está próxima al valor 0 si se representa el conocimiento como un porcentaje e indica que el usuario no conoce nada o muy poco sobre un determinado elemento (la inicialización por defecto del conocimiento del usuario se hace a este valor). De manera similar, la etiqueta “total” tendría un valor de porcentaje cercano a 100 y representa un conocimiento completo por parte del usuario.

Para facilitar la realización de operaciones matemáticas con grados de conocimiento, asociamos a cada etiqueta semántica, tal y como se muestra en la tabla 2, un valor numérico. En dicha tabla, también se muestra el porcentaje aproximado por cada grado de conocimiento.

Tabla 1: Def 8.4[Etiqueta semántica]

$\mathbf{et_{SEM}}$	valor-numérico	porcentaje
“nulo”	0	≈ 0%
“bajo”	1	≈ 25%
“medio”	2	≈ 50%
“alto”	3	≈ 75%
“total”	4	≈100%

[1] En la generación de rutas guiadas (capítulo 20) y para las comprobaciones de consistencia (capítulo 10), un mayor número de etiquetas incrementará la complejidad.

Def 8.5 [Función de conocimiento] Para referenciar el grado de conocimiento acerca de un elemento *x*, definimos una función de conocimiento, notada como ***K(x)***, que representa en valor numérico el grado de conocimiento acerca del ítem o concepto identificado por *x*.

El estado de conocimiento del usuario, al que de forma abreviada notamos aquí **estadoK**, se puede ver como un conjunto de parejas $<i_j, K(i_j)>$. Donde cada pareja especifica el grado de conocimiento, $K(i_j)$, que el usuario posee para cada uno de los ítems, i_j, del sistema hipermedia. De este modo, la ecuación 1 completa la definición de estado de conocimiento proporcionada en Def 8.2.

$$estadoK = \bigcup_{i_j \in I(EC_M)} \langle i_j, K(i_j) \rangle \text{ donde I}(EC_M) \text{ es el dominio de información de la } EC_M \quad (1)$$

Utilizando el estado de conocimiento del usuario acerca de los ítems de información, *estadoK*, y aplicando las reglas de peso (capítulo 11) es inmediato obtener un estado de conocimiento más completo, *estadoK''*, que incorpora de forma explícita el conocimiento del usuario acerca de los conceptos del dominio conceptual(C).

$$estadoK'' = \bigcup_{i_j \in I(EC_M)} \langle i_j, K(i_j) \rangle \cup \bigcup_{c_j \in C(EC_M)} \langle c_j, K(c_j) \rangle \quad (1')$$

No obstante, para la evaluación de las reglas de conocimiento es suficiente con *estadoK*.

3. DEFINICIÓN DE LAS REGLAS DE CONOCIMIENTO

En cada estructura conceptual de aprendizaje, el autor puede definir, si lo desea, un conjunto de reglas de conocimiento (Rk-autor) que le permiten expresar los prerrequisitos pedagógicos que son necesarios para acceder al contenido de cualquier ítem mostrado en dicha estructura.

Def 8.6 [Rk-autor 1] Dentro de una estructura conceptual de aprendizaje concreta notaremos **Rk-autor(i_j)** a una regla de conocimiento asociada al ítem i_j. Para diferenciar las distintas reglas de conocimiento definidas por el autor para un mismo ítem las numeramos con un superíndice, como sigue: Rk-autor^1(i_j), Rk-autor^2(i_j), Rk-autor^3(i_j),...

En una regla **Rk-autor^k(i_j)**, el autor formula las condiciones de conocimiento necesarias para comprender la información contenida en el ítem i_j, así como, para que esta visita sea productiva. Es posible definir distintas reglas de conocimiento sobre un mismo ítem porque pueden existir distintas combinaciones de conocimiento válidas, es decir, distintos caminos instructivos que preparan al usuario para asimilar fructíferamente un ítem de información.

Def 8.7 [Rk-autor 2] Notamos **Rk-autor(EC_A^i)** al conjunto de reglas de conocimiento definidas por el autor en una estructura conceptual de presentación específica, EC_A^i. Este conjunto pone de manifiesto las relaciones de aprendizaje existentes entre los distintos ítems de su dominio de información. Por lo tanto, las reglas del conjunto deben hacer referencia a ítems mostrados en la estructura conceptual de presentación sobre la que se define la EC_A^i.

La cardinalidad del conjunto Rk-autor(EC_A^i) no tiene por qué coincidir con el número de ítems mostrados en la presentación correspondiente, ya que un ítem puede tener asociadas cero, una o varias reglas de conocimiento.

3.1 Lenguaje para construir una Rk-autor

Las reglas de conocimiento son construidas por el autor usando un lenguaje cuya sintaxis y semántica se describen en esta sección.

3.1.1 Sintaxis del lenguaje

El lenguaje se compone de los siguientes elementos:

- Constantes:

Las constantes del lenguaje son las cinco etiquetas semánticas expresadas en valor numérico, esto es: 0, 1, 2, 3 y 4. También son constantes los identificadores de los ítems de información existentes en la presentación asociada a la EC_A. Ejemplos de identificadores serían I1, I2, I3,...

- Variables:

Cada variable será instanciada con alguna constante del lenguaje. Son variables con tipo, unas referencian ítems (i_j, i_k), otras valores de conocimiento (valor, val1, val2), etc.

- Predicados:

Existen ocho predicados disponibles para expresar una necesidad de conocimiento sobre un ítem de información (véase el diagrama de la figura 1). Atendiendo al número de argumentos, podemos distinguir tres grupos de predicados:

Grupo A. Los predicados Igual(i_j, valor), MenorE(i_j, valor), MenorNE(i_j, valor), MayorE(i_j, valor) y MayorNE(i_j, valor) requieren dos argumentos: la variable i_j que será instanciada con un ítem y la variable *valor* que será instanciada con una etiqueta semántica. Ejemplos de uso de estos predicados son Igual(I1, 3) o MenorNE(I3, 2).

Grupo B. Los predicados IntervaloE(i_j, val1, val2) e IntervaloNE(i_j, val1, val2) requieren tres argumentos: la variable i_j que será instanciada con un ítem y las variables *val1 y val2* que serán instanciadas con sendas etiquetas semánticas.

Grupo C. El predicado $OR_{SET}(i_j, \{predicado^1(i_j),..., predicado^n(i_j)\})$ tiene un número variable de argumentos. El primero, es siempre la variable i_j que será instanciada con un ítem. Después aparecerán dos o más predicados del lenguaje sobre el ítem i_j. Dichos predicados deben ser del grupo A o B. Una posible instanciación de este predicado sería OR_{SET}(I2, {MayorE(I2, 2), IntervaloNE(I2, 0, 1)}).

Además de los anteriores, existe un predicado especial Visitar(i_j) que requiere sólo un argumento, i_j, que será instanciado con un ítem de información.

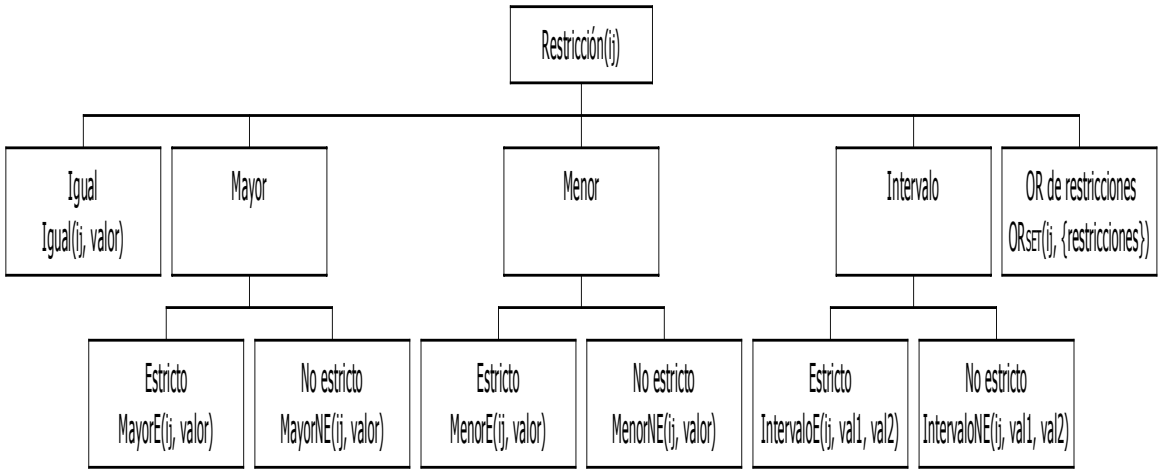


Figura 1: Predicados lógicos para restringir conocimiento

- Operadores lógicos: and.

Sólo se permite el operador *and* para combinar los predicados en una regla de conocimiento. De esta forma se pretende comprometer al autor a separar en reglas de conocimiento diferentes los recorridos pedagógicos distintos que preparan al usuario para visitar un ítem. Esta separación hace explícita la existencia de distintas formas de lograr un determinado conocimiento, y facilita posteriormente la evolución de las reglas al autor.

Por ejemplo, si el autor cree que para comprender el ítem I9 es necesario tener un conocimiento "alto" sobre I12 o bien tener un conocimiento mayor que "medio" sobre I3 e I5, es que existen dos caminos diferentes y como tales deben ser separados en dos reglas de conocimiento para I9.

- Fórmulas:

Una fórmula es una expresión del lenguaje que se obtiene de la aplicación de operadores a los predicados. Así, una fórmula simple es un predicado instanciado de forma adecuada. Y, una fórmula compleja es una combinación de predicados de conocimiento mediante el operador *and*.

- Regla o cláusula:

Las cláusulas del lenguaje tienen la forma: Cuerpo → Cabeza. Donde el *cuerpo* es una fórmula simple o compleja del lenguaje constituida por predicados de conocimiento (al menos uno) y la *cabeza* es siempre una formula simple formada únicamente por el predicado especial Visitar(i_j).

En la tabla 2 se muestra la especificación sintáctica del lenguaje en BNF extendido. Como puede verse, el autor puede crear fácilmente las reglas de conocimiento con el lenguaje proporcionado.

Tabla 2: EBNF del lenguaje para Rk-autor

EBNF del lenguaje para construir Rk-autor
<Rk-autor> ::= <cuerpo> → <cabeza>
<cabeza> ::= Visitar '(' <identificador_ítem>')'
<cuerpo> ::= <predicado_conocimiento> { and <predicado_conocimiento>}
<predicado_conocimiento> ::= <predicado_grupoA> | <predicado_grupoB> | <predicado_grupoC>
<predicado_grupoC> ::= OR_{SET} '(‘<identificador_ítem>, '{' (<predicado_grupoA> | <predicado_grupoB>) ',' (<predicado_grupoA> | <predicado_grupoB>) { ',' (<predicado_grupoA> | <predicado_grupoB>) } '}' ')'
<predicado_grupoA> ::= (MayorE | MenorE | MayorNE | MenorNE | Igual) '(' <identificador_ítem>, <etiqueta_semántica> ')'
<predicado_grupoB> ::= (IntervaloE | IntervaloNE) '(' <identificador_ítem>, <etiqueta_semántica>, <etiqueta_semántica> ')'
<etiqueta_semántica> ::= [0-4]
<identificador_ítem> ::= "El identificador de algún ítem existente en la EC_A"

3.1.2 Semántica del lenguaje

Es importante que destaquemos que la semántica que se asocia al lenguaje está en función del uso que deseamos hacer de éste para expresar restricciones de conocimiento sobre ítems de información. No obstante, la semántica de los predicados de conocimiento coincide con la que de forma clásica se ha atribuido a los operadores relacionales, sólo que en nuestro caso aplicada a grados de conocimiento. Del mismo modo, la semántica del predicado OR_{SET} se basa en el significado tradicional del predicado lógico *or*. Por su parte, la semántica del predicado *and* no ha sido redefinida. Todo esto contribuye a que el uso del lenguaje sea bastante intuitivo.

- Interpretación de la regla:

La valoración semántica del cuerpo de una regla de conocimiento se realiza a partir de la semántica de los predicados de conocimiento que lo componen. Concretamente, la interpretación del cuerpo es la combinación *and* de las interpretaciones obtenidas para los distintos predicados que lo forman. De este modo un cuerpo es interpretado como cierto (*true*), sólo cuando todos sus predicados lo son.

Para que el cuerpo de la regla sea semánticamente correcto es necesario que todos los predicados que lo componen lo sean. Pero además se exige que no contenga dos predicados sobre el mismo ítem. Esta condición es introducida para evitar que el autor exija la satisfacción de dos predicados de conocimiento incompatibles para el mismo ítem.

Por ejemplo la combinación MayorE(i_j, 3) and MenorE(i_j, 2) no es satisfacible por ningún estado de conocimiento y no debe ser permitida.

Por supuesto, tampoco es semánticamente correcto incluir en el cuerpo de la regla un predicado de conocimiento sobre el ítem de la cabeza. Ya que imponer una condición de conocimiento sobre un ítem para poder visitarlo, además de no tener demasiado sentido, supone un ciclo que seguramente convertiría en inaccesible a dicho ítem.

- Interpretación del predicado Visitar(i_j)

El predicado Visitar(i_j) representa la visita del ítem que se instancia en la variable i_j. Indicando que para dicha visita es para la que se imponen los requisitos de conocimiento incluidos en el cuerpo de la regla.

- Interpretación de un predicado de conocimiento:

La interpretación de un predicado se realiza instanciando sus variables. Dicha interpretación se hace sobre el estado de conocimiento de un usuario particular. La semántica de cada predicado es definida en la tabla 3. En la última columna se indica qué condición o condiciones de conocimiento deben satisfacerse sobre el ítem del predicado para que éste sea interpretado como cierto.

Tabla 3: Interpretación de un predicado de conocimiento

Predicado		**Cierto SI**
Igual(i_j, valor)		$K(i_j)$ = valor
Mayor	MayorE(i_j, valor)	$K(i_j) >$ valor
	MayorNE(i_j, valor)	$K(i_j) \geq$ valor
Menor	MenorE(i_j, valor)	$K(i_j) <$ valor
	MenorNE(i_j, valor)	$K(i_j) \leq$ valor
Intervalo	IntervaloE(i_j,val1,val2)	val1 $< K(i_j) <$ val2
	IntervaloNE(i_j,val1,val2)	val1 $\leq K(i_j) \leq$ val2
$OR_{SET}(i_j,\{predicado^1(i_j),...,predicado^n(i_j)\})$		$K(i_j)$ satisface alguna de los *n* predicados del conjunto.

Como se deduce de su semántica, el último predicado, OR_{SET}, es en realidad, una combinación *or* de predicados definidos sobre el mismo ítem. Este predicado se incluye con la intención de permitir únicamente combinaciones *or* de predicados instanciados con el mismo ítem. Aunque es prescindible, puede resultarle cómodo al autor para expresar en una misma regla varias posibilidades de conocimiento sobre un ítem. Es una concesión en la separación total de caminos pedagógicos en favor de la comodidad del autor.

Para que los predicados sean correctos desde un punto de vista semántico es necesario que el autor satisfaga una serie de condiciones durante su instanciación. Estas restricciones se muestran en la tabla 4 y tienen como objetivo evitar que el autor exprese un requisito de conocimiento imposible.

Por ejemplo, el predicado MayorE(I9, 4) es correcto sintáctica, pero no semánticamente, ya que exige sobre el ítem I9 un grado de conocimiento mayor que 4, lo cual es

imposible porque 4 representa ya el conocimiento máximo, esto es "total" (véase la tabla 1).

Para hacerlo de forma general, se utiliza la variable v_max que representa el valor de conocimiento más alto (variará en función del número de etiquetas semánticas utilizadas), en nuestro caso 4. Siempre que no se diga nada en contra, el valor debe estar comprendido entre 0 y v_max, esto es: $0 \leq valor \leq v\_max$.

Tabla 4: Restricciones semánticas en la construcción de un predicado de conocimiento

Predicado		**Restricciones**
Igual(i_j, valor)		_
Mayor	MayorE(i_j, valor)	valor < v_max
	MayorNE(i_j, valor)	_
Menor	MenorE(i_j, valor)	valor > 0
	MenorNE(i_j, valor)	_
Intervalo	IntervaloE(i_j,val1,val2)	val1 < v_max -1 val2 > 1 val2 > val1+1
	IntervaloNE(i_j,val1,val2)	val2 ≥ val1
$OR_{SET}(i_j,\{predicado^1(i_j),...,predicado^n(i_j)\})$		Todos los predicados del conjunto deben implicar al ítem i_j. Para evitar redundancia, no deben coexistir en el conjunto OR_{SET} dos predicados con idéntica semántica. Dos predicados tienen la misma semántica si se hacen ciertos para los mismos valores de $K(i_j)$. Este es el caso de las siguientes parejas de predicados: - MayorE(i_j,val) y MayorNE(i_j, val+1), - MenorE(i_j, val) y MenorNE(i_j, val-1) e - IntervaloE(i_j,val1, val2) y IntervaloNE(i_j, val1-1, val2+1).

Como puede verse en la tabla 4, no son muchas las restricciones semánticas que el autor debe cumplir en la utilización de los predicados de conocimiento y la mayoría de ellas son elementales. Esto refuerza, nuevamente, la simplicidad del lenguaje de autor, para el que en la sección 3.3 se muestra una posible interfaz de autor.

3.2 Completitud y consistencia del lenguaje para Rk-autor

En la tabla 5 se muestra la completitud del lenguaje proporcionado al autor para definir las Rk-autor. Para ello, en la primera columna de la tabla se han colocado todas las combinaciones de restricciones de conocimiento semánticamente válidas sobre un

mismo ítem. Los predicados utilizados para definir estas restricciones son los cinco operadores relaciones utilizados tradicionalmente: > (mayor), < (menor), = (igual), <= (menor o igual) y >= (mayor o igual).

La última columna de la tabla muestra como cada combinación de restricciones puede ser expresada mediante uno sólo de los predicados de conocimiento proporcionados al autor (figura 1). De esta forma se demuestra que exigir un único predicado sobre un ítem en el cuerpo de una regla Rk-autor no merma la capacidad expresiva del lenguaje, al contrario evita inconsistencias y redundancia.

En la segunda columna se indican las condiciones de instanciación que deben satisfacerse para que la combinación de restricciones sea semánticamente válida.

Por ejemplo el conocimiento sobre un ítem no puede ser a la vez igual a 3 y menor que 2, porque 3 es mayor que 2.

En la tercera columna de la tabla puede comprobarse, además, cómo la semántica de la combinación coincide fielmente con la semántica del predicado equivalente (véase la tabla 3). Por supuesto, para obtener estas equivalencias o simplificaciones se asume que se satisfacen las condiciones de validez establecidas en la segunda columna.

Tabla 5: Completitud del lenguaje Rk-autor

Combinación	Válido Si	Equivale a	Predicado
Combinaciones posibles con AND			
$K(i_j)=v1$ and $K(i_j)=v2$	$v1=v2$	$K(i_j) = v1$	Igual(i_j, v1)
$K(i_j)=v1$ and $K(i_j)>v2$	$v1>v2$	$K(i_j) = v1$	Igual(i_j, v1)
$K(i_j)=v1$ and $K(i_j)<v2$	$v1< v2$	$K(i_j) = v1$	Igual(i_j, v1)
$K(i_j)=v1$ and $K(i_j) \geq v2$	$v1 \geq v2$	$K(i_j) = v1$	Igual(i_j, v1)
$K(i_j)=v1$ and $K(i_j) \leq v2$	$v1 \leq v2$	$K(i_j) = v1$	Igual(i_j, v1)
$K(i_j)>v1$ and $K(i_j)>v2$	_	$K(i_j) > \max(v1,v2)$	MayorE(i_j,max(v1,v2))
$K(i_j)<v1$ and $K(i_j)<v2$	_	$K(i_j) < \min(v1,v2)$	MenorE(i_j,min(v1,v2))
$K(i_j)\geq v1$ and $K(i_j)\geq v2$	_	$K(i_j) \geq \max(v1,v2)$	MayorNE(i_j,max(v1,v2))
$K(i_j)\leq v1$ and $K(i_j)\leq v2$	_	$K(i_j)\leq \min(v1,v2)$	MenorNE(i_j,min(v1,v2))
$K(i_j)>v1$ and $K(i_j)\geq v2$	_	$v1>v2 \equiv K(i_j)>v1$	MayorE(i_j,v1)
		$v2\geq v1 \equiv K(i_j)\geq v2$	MayorNE(i_j,v2)
$K(i_j)<v1$ and $K(i_j)\leq v2$	_	$v1<v2 \equiv K(i_j) < v1$	MenorE(i_j,v1)

		$v2 \leq v1 \equiv K(i_j) \leq v2$	MenorNE(i_j,v2)
$K(i_j)>v1$ and $K(i_j)<v2$	$v2>v1+1$	$v1<K(i_j)< v2$	IntervaloE (i_j,v1,v2)
$K(i_j)\geq v1$ and $K(i_j)\leq v2$	$v2\geq v1$	$v1\leq K(i_j)\leq v2$	IntervaloNE (i_j,v1,v2)
$K(i_j)>v1$ and $K(i_j)\leq v2$	$v2>v1$	$v1+1\leq K(i_j)\leq v2$	IntervaloNE (i_j,v1+1,v2)
$K(i_j)\geq v1$ and $K(i_j)<v2$	$v2>v1$	$v1\leq K(i_j)\leq v2-1$	IntervaloNE (i_j,v1,v2-1)
Combinaciones posibles con OR			
Todas las combinaciones de restricciones sobre un mismo ítem i_j conectadas mediante el operador lógico *or* se pueden representar mediante el predicado $OR_{SET}(i_j$, {conjunto de restricciones}).			

La demostración de que cualquier combinación de restricciones de conocimiento sobre un ítem puede representarse con un sólo predicado de conocimiento se apoya en las siguientes premisas:

a) Una combinación de restricciones compuesta por dos restricciones simples sobre un mismo ítem y unidas mediante el operador lógico *and,* se puede representar mediante el operador Intervalo (estricto o no estricto), siempre y cuando, una sea de tipo mayor (> o ≥), la otra sea de tipo menor (< o ≤) y la combinación de ambas sea válida.

Por ejemplo, la combinación $K(i_j)>0$ and $K(i_j) \leq 3$ puede ser expresada con el predicado IntervaloNE (i_j,1,3).

b) Una combinación de restricciones compuesta por dos restricciones simples sobre un mismo ítem y conectadas mediante el operador *and*, tal que las restricciones implicadas no satisfacen la condición expuesta en a) puede simplificarse eliminando la restricción menos restrictiva.

Por ejemplo, la combinación $K(i_j)>0$ and $K(i_j)>2$ puede ser expresada con el predicado MayorE (i_j,2).

Cuando una de las restricciones implicadas es de igualdad (=), la simplificación tiene sentido únicamente si el valor constante al que se iguala satisface la otra restricción, que en dicho caso será eliminada.

Por ejemplo, la combinación $K(i_j)=3$ and $K(i_j)>2$ tiene sentido porque 3 es mayor que 2, y sólo es cierta cuando i_j es igual a 3, por lo tanto puede ser expresada mediante el predicado Igual(i_j, 3).

c) Si la combinación de restricciones está compuesta por más de dos restricciones simples unidas mediante el operador *and*, se aplican las premisas a) y b), según corresponda, de forma repetida hasta reducir la combinación a un único predicado.

Primero se aplican las simplificaciones de tipo b) mientras que sea posible, y después, si es necesario se aplica la simplificación de tipo a).

Por ejemplo, la combinación de cuatro restricciones $K(i_j)\geq 0$ and $K(i_j)\geq 1$ and $K(i_j)\leq 3$ and $K(i_j) \leq 4$ puede ser reducida a $K(i_j)\geq 1$ and $K(i_j) \leq 3$ aplicando dos simplificaciones de tipo b) y después expresada siguiendo a) con el predicado IntervaloNE(i_j, 1, 3).

d) Por último, el predicado OR_{SET} permite representar cualquier combinación de restricciones simples unidas mediante el operador lógico *or,* donde todas las restricciones afectan al mismo ítem.

Por ejemplo, la combinación $K(i_j)>2$ or $K(i_j)=3$ es equivalente al predicado OR_{SET} (i_j, {MayorE(i_j,2), Igual(i_j, 3)}).

Tal y como se ha expuesto, con un sólo predicado es posible exigir cualquier restricción de conocimiento sobre un ítem, evitando además las posibles inconsistencias que pudieran surgir en caso de combinar incorrectamente varios predicados (véase tabla 6).

Tabla 6: Valores que hacen inconsistente una combinación de predicados

Combinación	Inconsistente si
MayorE(i_j, val1) and Igual(i_j, val2)	val2 ≤ val1
MayorNE(i_j, val1) and Igual(i_j, val2)	val2 < val1
MenorE(i_j, val1) and Igual(i_j, val2)	val2 ≥ val1
MenorNE(i_j, val1) and Igual(i_j, val2)	val2 > val1
MayorE(i_j, val1) and MenorE(i_j, val2)	val2 ≤ val1+1
MayorE(i_j, val1) and MenorNE(i_j, val2)	val2 ≤ val1
MayorNE(i_j, val1) and MenorE(i_j, val2)	val2 ≤ val1
MayorNE(i_j, val1) and MenorNE(i_j, val2)	val2 < val1

Partiendo de que cada predicado tiene asociadas una serie de restricciones semánticas que garantizan la ausencia de contradicción en el mismo y de que en el cuerpo de una regla de conocimiento no puede haber dos predicados sobre un mismo ítem, queda garantizada la consistencia de las reglas de conocimiento definidas por el autor desde su propio proceso de creación. Las restricciones asociadas a las acciones evolutivas que se proporcionan al autor para definir y modificar las Rk-autor son las que permiten garantizar las citadas restricciones de consistencia (véase la sección 6).

3.3 Interfaz para construir Rk-autor

En resumen, para restringir el grado de conocimiento sobre un ítem, el autor puede utilizar ocho predicados lógicos diferentes (figura 1), siempre y cuando satisfaga las restricciones, sintácticas y semánticas, impuestas para la utilización de los mismos (tablas 2 y 4). El operador intervalo y el operador OR_{SET} eliminan la necesidad de tener varios predicados sobre un mismo ítem en el cuerpo de una regla de conocimiento. Para lograr este objetivo, ambos predicados engloban varias restricciones de conocimiento sobre el mismo ítem en una única restricción.

Para crear las reglas de conocimiento resulta muy adecuado suministrar al autor una interfaz gráfica que permita una interacción flexible e intuitiva. La figura 2 muestra una propuesta, que incorpora una barra horizontal (situada en la parte superior de la ventana) con todos los predicados disponibles para restringir el conocimiento sobre un ítem y una barra vertical (ubicada a la izquierda) que muestra el valor numérico equivalente a cada etiqueta semántica.

De esta forma, el autor sólo tiene que seleccionar el ítem al que se asocia la regla (se marca en color rojo oscuro) y después ir seleccionando los ítems que formarán parte del cuerpo de dicha regla (se marcan en color verde claro). Para establecer una restricción sobre un ítem resaltado en verde, basta con arrastrar el predicado que se desea desde la barra superior hasta él, y rellenar el recuadro que aparece con los valores numéricos requeridos. En el caso de que se asocie más de un predicado a un ítem automáticamente se considera un OR_{SET} (todos los predicados se engloban dentro de un recuadro con línea discontinua).

Así, la regla de conocimiento definida por el autor en la figura 2 sería la siguiente: IntervaloE(I1,1,4) and OR_{SET}(I2, {MayorE(I2,3), MenorNE(I2,1)}) → Visitar (I4).

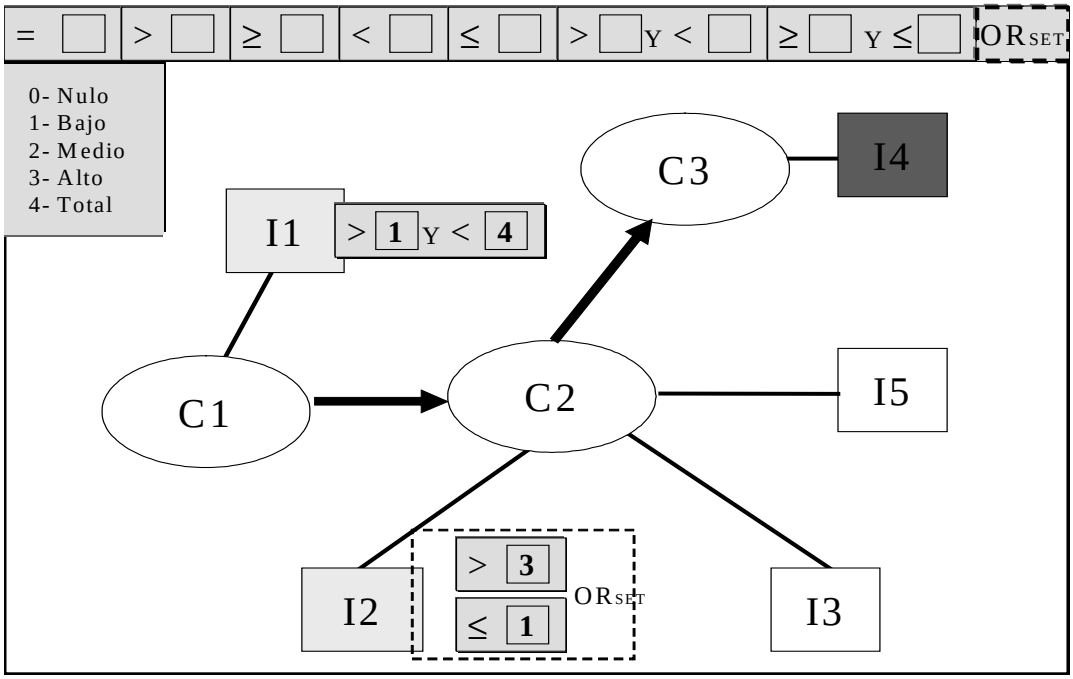


Figura 2: Interfaz gráfica para una regla de conocimiento

4. REPRESENTACIÓN INTERNA DE LAS REGLAS DE CONOCIMIENTO

Antes de especificar sintáctica y semánticamente el lenguaje utilizado para representar internamente las reglas de conocimiento, es necesario aclarar y justificar por qué es necesaria esta representación interna.

Si nos fijamos en las restricciones de conocimiento que podemos expresar con los predicados Mayor y Menor, vemos que existe una importante diferencia entre ellos. Si asumimos que conforme navega el usuario, su estado de conocimiento se mantiene o aumenta como consecuencia de la información que consulta, podemos establecer que:

a) una restricción de conocimiento de tipo Mayor, una vez que es satisfecha por el usuario va a ser satisfecha siempre en adelante,

b) mientras que una restricción de tipo Menor, es satisfecha inicialmente y deja de serlo en el momento que el grado de conocimiento del usuario supera el umbral exigido en la restricción.

El caso del predicado Igual es similar al del predicado Menor (b), salvo que ahora únicamente un grado de conocimiento concreto satisface la restricción de igualdad. De modo que ni antes de alcanzar ese grado, ni al aumentarlo se satisface ya dicha restricción.

Por su parte, en los predicados de tipo Intervalo se integran ambos tipos de restricciones (a y b), ya que requieren un grado de conocimiento mínimo y máximo sobre un ítem.

La diferencia de estas restricciones es importante, y deben ser separadas por dos motivos:

Motivo 1. El objetivo perseguido con cada una de estas restricciones es distinto. Las primeras (a) pretenden conseguir que el usuario esté preparado para comprender la información contenida en un ítem. Y por ello exigen un conocimiento mínimo sobre algunos ítems necesarios. Por su parte, las segundas (b) quieren evitar que el usuario pierda el tiempo leyendo un ítem muy simple o redundante para él, por lo que fijan un conocimiento máximo sobre otros ítems relacionados.

Motivo 2. El primer tipo de restricciones puede ser usado para restringir el acceso a los ítems que el usuario no es capaz de asimilar. Sin embargo, el segundo tipo no puede ser utilizado en este sentido. Primero porque no parece muy sensato impedir al usuario la visita de un ítem porque sea demasiado simple o redundante, ya que puede ser su deseo repasarlo o consultar algún dato incluido en éste. Y segundo, porque en caso de hacerlo no habría forma de garantizar que navegando una EC_A el usuario pueda alcanzar conocimiento "total" sobre todos los ítems mostrados en dicha estructura. Incluso, podría ocurrir que determinados ítems se convirtiesen en inalcanzables, como consecuencia de que una restricción de conocimiento de tipo b) deje de satisfacerse. Por este motivo, estas restricciones deben únicamente desaconsejar la visita a un ítem, pero no prohibirla.

Para ilustrar el segundo motivo, se plantea una situación que tendría lugar si las restricciones de tipo b) fuesen utilizadas para prohibir el acceso a un ítem.

Supongamos que el autor establece en la única regla de conocimiento de I9 una restricción de tipo b) que exige un conocimiento menor que "medio" sobre I7.

$$Rk(I9): MenorE(I7, 2) \rightarrow Visitar(I9)$$

En el momento en que el conocimiento del usuario sobre I7 sea mayor o igual que "medio", el ítem I9 se convierte en inaccesible. Supongamos que en ese momento el conocimiento del usuario sobre I9 es "nulo" y que acceder a su contenido es la única forma de obtener conocimiento acerca de I9, entonces el conocimiento máximo que ese usuario puede alcanzar sobre I9 es "nulo".

En las reglas definidas por el autor no existe separación entre las restricciones de tipo (a) y (b). Sin embargo, internamente es conveniente que ésta exista de manera explícita, ya que las restricciones de tipo (a) son utilizadas para determinar si es posible acceder a un ítem (**restricciones de accesibilidad**). Mientras que las de tipo (b) se usan para evaluar la idoneidad de una visita (**restricciones de idoneidad**).

Internamente, una restricción de autor estará compuesta por una o varias restricciones de accesibilidad y/o idoneidad. Aunque la representación interna de las reglas es distinta a la representación del autor, tal y como se demuestra en secciones posteriores, la equivalencia entre ambas es total.

4.1 Sintaxis del lenguaje para Rk

Notamos Rk a la representación interna de las reglas de conocimiento. Siendo $Rk(i_j)$ el conjunto de reglas de conocimiento definidas para el ítem i_j. Y numerando las distintas reglas de conocimiento asociadas internamente a un mismo ítem como $Rk^1(i_j)$, $Rk^2(i_j)$,..., $Rk^m(i_j)$.

- Constantes del lenguaje:

Las cinco etiquetas semánticas expresadas en valor numérico (0, 1, 2, 3 y 4). Y los identificadores de los ítems de información existentes en la presentación sobre la que se define la EC_A.

- Variables del lenguaje:

Cada variable será instanciada con alguna constante del lenguaje. Son nombres de variables: i_j, val,...

- Predicados:

Existen dos tipos distintos de predicados para expresar restricciones de conocimiento.

> **Def 8.8 [Restricción de accesibilidad]** Una restricción de accesibilidad, a la que llamamos $Restricción^A$, establece cuál es el grado de conocimiento mínimo que debe poseer el usuario sobre un ítem para que el acceso a otro determinado ítem esté *permitido*. El predicado para especificar una restricción de este tipo es de la forma: $K(i_j) \geq val$, donde la variable i_j será instanciada con el identificador de un ítem de información y la variable *val* con una etiqueta semántica.

Def 8.9 [Restricción de idoneidad] Una restricción de idoneidad, notada como Restricción^{I}, determina cuál es el grado de conocimiento máximo o exacto que debe poseer el usuario sobre un ítem para que la visita a otro determinado ítem sea *aconsejable*. Existen dos predicados para expresar una restricción de este tipo: $K(i_j) \leq val$ y $K(i_j) = val$. En ambos casos, i_j se instancia con un ítem y *val* con una etiqueta de conocimiento.

Existe un predicado especial para representar la visita del ítem sobre el que se decide la accesibilidad e idoneidad. La sintaxis de dicho predicado es Visitar(i_j), siendo i_j el identificador de un ítem de información.

- Operadores lógicos: and y *and‴*

And es el operador de la lógica clásica. Mientras que *and‴* es un nuevo operador lógico cuya semántica se define en la siguiente sección (véase la tabla 9).

- Fórmulas del lenguaje:

True es una fórmula atómica.

Una *fórmula simple de accesibilidad* es la instanciación del predicado de accesibilidad ≥. Del mismo modo que una *fórmula simple de idoneidad* es la instanciación de un predicado de idoneidad (= o ≤).

Existen también dos tipos de formulas complejas. Una *fórmula compleja de accesibilidad* está compuesta por la combinación de dos o más fórmulas simples de accesibilidad mediante el operador lógico *and*. Una *fórmula compleja de idoneidad* es aquella que conecta mediante el operador *and* un conjunto de formulas simples de idoneidad (al menos dos).

- Regla o cláusula:

Una regla se compone de un cuerpo o antecedente y una cabeza o consecuente: cuerpo → cabeza. Los predicados lógicos del *cuerpo* representan restricciones de conocimiento que serán evaluadas para determinar el valor que toma la *cabeza*.

El cuerpo o antecedente se divide en dos partes: accesibilidad e idoneidad, unidas mediante el operador lógico *and‴*. El **antecedente de accesibilidad** puede ser una fórmula atómica (*true*), una fórmula de accesibilidad simple o una fórmula de accesibilidad compleja. Por su parte, el **antecedente de idoneidad** puede ser *true* o una fórmula de accesibilidad, ya sea simple o compleja.

La cabeza o consecuente está formada únicamente por el predicado especial Visitar(i_j).

En la tabla 7 se define usando EBNF la sintaxis del lenguaje utilizado para representar internamente las reglas de conocimiento.

Tabla 7: EBNF del lenguaje para Rk

EBNF del lenguaje para construir Rk
<Rk> ::= <cuerpo> → <cabeza>

<cabeza> ::= Visitar '(' <identificador_ítem>')'
<cuerpo> ::= <antecedente_accesibilidad> *and‴* <antecedente_idoneidad>
<antecedente_accesibilidad> ::= true | <predicado_accesibilidad> {and <predicado_accesibilidad>}
<antecedente_idoneidad> ::= true | <predicado_idoneidad> {and <predicado_idoneidad>}
<predicado_accesibilidad> ::= K'('<identificador_ítem>')' '≥' <etiqueta_semántica>
<predicado_idoneidad> ::= K'('<identificador_ítem>')' ('≤' | '=') <etiqueta_semántica>
<etiqueta_semántica> ::= [0-4]
<identificador_ítem> ::= "El identificador de algún ítem existente en la EC_A"

4.2 Semántica del lenguaje para Rk

- Interpretación de un predicado:

La interpretación de un predicado consiste en evaluar la instanciación de éste sobre un determinado estado de conocimiento. Como resultado se obtendrá *true* o *false* según se satisfaga o no la condición de conocimiento impuesta en el estado evaluado.

No existen restricciones semánticas en la instanciación de los predicados.

- Interpretación de los antecedentes del cuerpo de la regla:

Si el antecedente es la fórmula atómica *true* la interpretación del antecedente es *true*. Si el antecedente es una fórmula simple la interpretación del antecedente coincide con la interpretación del predicado involucrado. Si por el contrario, el antecedente es una fórmula compleja, la interpretación del antecedente se obtiene combinando mediante el operador *and* la interpretación de todos los predicados que lo forman. De este modo, el antecedente se interpreta como *true* sólo si ésta es la interpretación de todos sus predicados.

Para que cualquiera de los dos antecedentes, de accesibilidad o de idoneidad, sea semánticamente correcto es necesario que se cumplan dos condiciones:

1. No pueden existir dos predicados en un mismo antecedente que afecten al mismo ítem.

2. No puede definirse en el antecedente un predicado que afecte al ítem de la cabeza de la regla.

En la tabla 8 se demuestra que no supone pérdida de expresividad alguna impedir que coexistan en un mismo antecedente dos predicados sobre el mismo ítem. Ya que cualquier combinación válida de dos predicados del mismo tipo (accesibilidad o idoneidad) puede ser expresada mediante un único predicado de ese tipo.

Tabla 8: Completitud del conjunto de predicados para restringir el conocimiento

Combinación de restricciones sobre i_j		Válido sólo si	Equivale a
Accesibilidad	$K(i_j) \geq v1$ and $K(i_j) \geq v2$	_	$K(i_j) \geq \max(v1,v2)$
Idoneidad	$K(i_j) = v1$ and $K(i_j) = v2$	$v1 = v2$	$K(i_j) = v1$
	$K(i_j) = v1$ and $K(i_j) \leq v2$	$v1 \leq v2$	$K(i_j) = v1$
	$K(i_j) \leq v1$ and $K(i_j) \leq v2$	_	$K(i_j) \leq \min(v1,v2)$

- Interpretación de la regla:

La interpretación del cuerpo de la regla determina el valor que toma el predicado de la cabeza, Visitar(i_j). Esta interpretación debe indicar si la visita del ítem i_j está permitida y en ese caso si es aconsejable.

Así, el predicado Visitar(i_j) es trivaluado, pudiendo tomar cualquiera de estos tres valores: "prohibido", "permitido" y "aconsejado". El valor que posee en cada momento se obtiene combinando la interpretación de los dos antecedentes del cuerpo de la regla mediante el operador lógico *and‴*. Este operador a partir de dos valores lógicos pertenecientes al conjunto {*true*, *false*} obtiene un valor lógico perteneciente al conjunto {"prohibido", "permitido", "aconsejado"}.

Una vez interpretado cada antecedente por separado, los valores de verdad obtenidos se combinan mediante el operador lógico *and‴* cuya semántica se define en la tabla 9.

Tabla 9: Semántica del operador *and‴*

antecedente_accesibilidad *and‴* antecedente_idoneidad		Interpretación de la regla Visitar(i_j)
Interpretación del antecedente_accesibilidad	Interpretación del antecedente_idoneidad	
true	true	Aconsejado
true	false	Permitido
false	true	Prohibido
false	false	Prohibido

Resumiendo, si el antecedente de accesibilidad es falso la regla de conocimiento prohíbe el acceso al ítem. En otro caso, cuando el antecedente de accesibilidad es cierto, el acceso a dicho ítem está permitido, pero es aconsejable sólo en el caso de que el antecedente de idoneidad sea también cierto.

- Interpretación de un conjunto de reglas:

Cuando existen varias reglas de conocimiento $Rk^1(i_j)$, $Rk^2(i_j)$, ..., $Rk^m(i_j)$ asociadas al mismo ítem, basta con que una de ellas se resuelva con el valor "aconsejado" para que la visita al ítem se considere aconsejable. En caso de que ninguna de las reglas

asociadas se evalúe con dicho valor es suficiente que en una de ellas se obtenga el valor "permitido" para que el ítem sea accesible y el usuario tenga la capacidad de visitarlo. Sólo cuando todas las reglas asociadas se evalúan con valor "prohibido" el usuario tiene prohibido el acceso al ítem. Por tanto, existe una combinación implícita de todas las reglas de conocimiento con idéntica cabeza. Así dado el siguiente conjunto de reglas:

$Rk^1(i_j)$: $cuerpo^1 \rightarrow Visitar(i_j)$
$Rk^2(i_j)$: $cuerpo^2 \rightarrow Visitar(i_j)$
.....
$Rk^m(i_j)$: $cuerpo^m \rightarrow Visitar(i_j)$

La combinación de todas ellas en una sola se realiza como se muestra en la ecuación 2:

$$Rk(i_j): cuerpo^1 \ or''' \ cuerpo^2 \ or''' \ ... \ or''' \ cuerpo^m \rightarrow Visitar(i_j) \qquad (2)$$

La semántica del operador lógico *or‴* que se utiliza para combinar las reglas de conocimiento asociadas a un mismo ítem, difiere de la semántica del *or* clásico puesto que trabaja con tres valores lógicos {"aconsejado", "permitido" y "prohibido"} en lugar de con dos. Dicha semántica se especifica en la tabla 10.

Tabla 10: Semántica del operador lógico *or‴*

valor1	valor2	**valor1** *or‴* **valor2**
Aconsejado	Aconsejado	Aconsejado
Aconsejado	Permitido	Aconsejado
Aconsejado	Prohibido	Aconsejado
Permitido	Permitido	Permitido
Permitido	Prohibido	Permitido
Prohibido	Prohibido	Prohibido

4.3 Árbol de accesibilidad

Las restricciones de acceso especificadas en las reglas de conocimiento $Rk^1(i_j)$, $Rk^2(i_j)$, ..., $Rk^m(i_j)$ asociadas a un ítem i_j, pueden ser representadas mediante un árbol de accesibilidad, al que notamos de forma abreviada $Tree_K(i_j)$.

Def 8.10 [Árbol de accesibilidad] El árbol de accesibilidad asociado a un ítem i_j, esto es $Tree_K(i_j)$, representa mediante una estructura jerárquica, las restricciones de conocimiento que el usuario debe satisfacer para poder acceder convenientemente al ítem i_j. El árbol integra las restricciones de conocimiento presentes en los antecedentes de accesibilidad de todas las reglas de conocimiento definidas para i_j.

La construcción del árbol de accesibilidad permite visualizar de forma clara los prerrequisitos pedagógicos necesarios para acceder a un ítem. Además esta representación en árbol es usada para identificar los ítems deseables durante la adaptación a la meta en la navegación por conocimiento (capítulos 18 y 19).

Para cada ítem, el sistema es capaz de generar de forma automática su árbol de accesibilidad. A continuación se definen los pasos principales del algoritmo que se utiliza para construir el $Tree_K(i_j)$ partiendo del conjunto de reglas de conocimiento $Rk^1(i_j)$, $Rk^2(i_j)$, ..., $Rk^m(i_j)$.

Paso 1. El nodo raíz en el árbol $Tree_K(i_i)$ es el ítem afectado por el predicado *Visitar(i_i)* en la cabeza de las reglas $Rk^1(i_i)$, $Rk^2(i_i)$, ..., $Rk^m(i_i)$, esto es, el ítem i_i al que se asocian dichas reglas de conocimiento.

Paso 2. Cada regla de conocimiento $Rk^k(i_i)$ donde el antecedente de accesibilidad no es vacío genera un subárbol colgando del ítem raíz i_j.

Paso 3. Todos los subárboles se conectan mediante una línea horizontal discontinua que modeliza un *or* clásico; ya que basta que se interprete como *true* el antecedente de accesibilidad de una sola de las reglas asociadas a i_i para que la visita al ítem esté permitida.

Paso 4. El subárbol de la regla de conocimiento $Rk^k(i_i)$ tiene tantos nodos como restricciones de conocimiento de tipo Restricción^A ($K(i_f) \geq val_f$) existan en el antecedente de accesibilidad de dicha regla.

Paso 4.1 Cada nodo del subárbol es etiquetado con el identificador del ítem implicado en la restricción, esto es i_f.

Paso 4.2 La rama que llega hasta el nodo i_f se etiqueta con el grado de conocimiento exigido para dicho ítem en la restricción, esto es $\geq val_f$.

Paso 4.3 Para representar la necesidad de que todas las restricciones del antecedente de accesibilidad sean satisfechas, se conectan todas las ramas del subárbol mediante una línea horizontal continua que modela un operador *and* clásico.

Algoritmo 1. Construcción del árbol de accesibilidad

Por ejemplo, supongamos que existen dos reglas de conocimiento $Rk^1(I3)$ y $Rk^2(I3)$ estableciendo la accesibilidad e idoneidad de la visita al ítem I3.

$Rk^1(I3)$: $[K(I1) \geq 2$ and $K(I2) \geq 3]$ *and‴* $K(I4) \leq 3 \rightarrow$ Visitar(I3)

$Rk^2(I3)$: $[K(I2) \geq 1$ and $K(I4) \geq 3$ and $K(I6) \geq 3]$ *and‴* true $\rightarrow$ Visitar(I3)

Ejecutando los pasos 1, 2 y 3 del algoritmo se obtiene para el ítem I3 un árbol de accesibilidad con la estructura que se muestra en la figura 3.

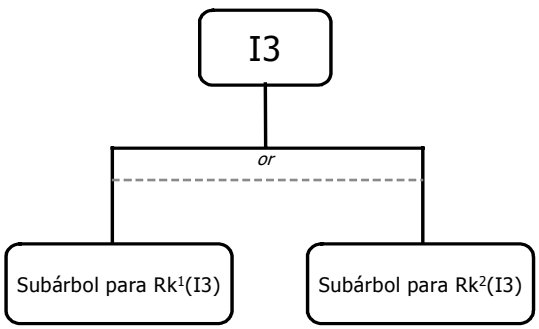


Figura 3: Primer nivel del $Tree_K(I3)$

En las figuras 4.1 y 4.2 se muestran los subárboles que se obtienen durante el paso 4 del algoritmo para las reglas de conocimiento $Rk^1(I3)$ y $Rk^2(I3)$ respectivamente. Con objeto de facilitar la comprensión, en el etiquetado de las ramas se han utilizado las etiquetas semánticas equivalentes a los valores numéricos empleados en la representación interna de las reglas.

En la figura 4.1 puede observarse que el antecedente de idoneidad $K(I4) \leq 3$ en $Rk^1(I3)$ es ignorado para la construcción del subárbol de accesibilidad.

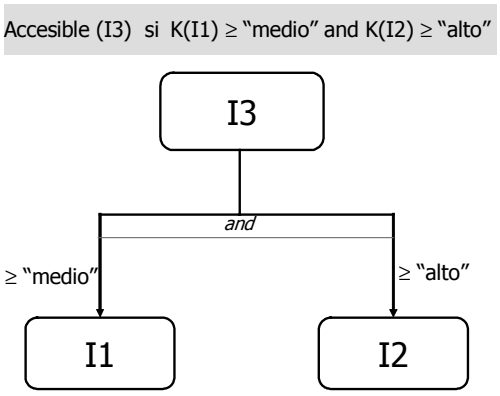


Figura 4.1: Subárbol para $Rk^1(I3)$

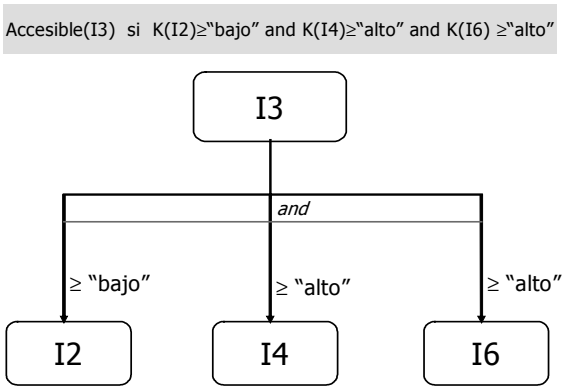


Figura 4.2: Subárbol para $Rk^2(I3)$

Como ocurre en el ejemplo con I2, a veces un mismo ítem aparece en varios subárboles del $Tree_K(i_j)$. Esto ocurre cuando dicho ítem es requerido en el antecedente de accesibilidad de varias reglas de conocimiento asociadas a i_j. Esta situación puede ser resuelta representado el ítem con un único nodo en el árbol de accesibilidad $Tree_K(i_j)$, de modo que a éste llegue una rama procedente de cada subárbol en el que aparezca.

En la figura 5 se muestra el árbol de accesibilidad finalmente obtenido para el ítem I3. El $Tree_K(I3)$ constituye una representación gráfica, muy clara, de los requisitos de conocimiento necesarios para comprender la información contenida en I3. De un simple vistazo se perciben las dos alternativas para acceder a I3: o bien se conocen con el grado indicado I1 e I2 o bien I2, I4 e I6.

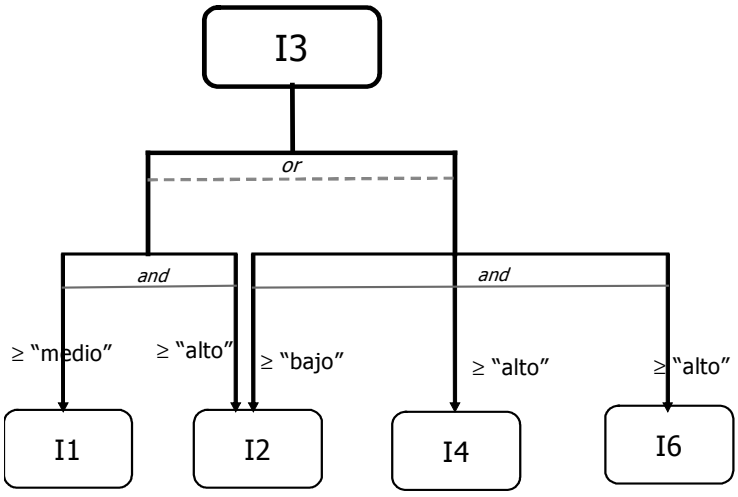


Figura 5: $Tree_K$ (I3)

Evaluando las restricciones asociadas a los nodos del último nivel del $Tree_K$ sobre el estado de conocimiento actual del usuario, es posible determinar si el ítem raíz es accesible para este usuario según los requisitos pedagógicos impuestos por el autor en las reglas de conocimiento.

5. CORRESPONDENCIA ENTRE RK-AUTOR Y RK

El sistema proporciona al autor ocho predicados lógicos que le permiten escribir las reglas de conocimiento con bastante expresividad y sin tener que preocuparse de hacer una separación explícita de las restricciones de accesibilidad y las restricciones de idoneidad. Es el sistema quién, posteriormente y de forma automática, obtiene a partir de las reglas de conocimiento de autor un conjunto equivalente de reglas con la representación que internamente se necesita. El procedimiento que se sigue para ello es el que se detalla en el siguiente algoritmo:

Paso 1. Dada una regla de conocimiento Rk-autor(i_j) el sistema genera una regla de conocimiento interna Rk(i_j) donde los predicados lógicos de autor: Igual, MayorE, MayorNE, MenorE, MenorNE, IntervaloE e IntervaloNE son expresados usando los predicados propios de la representación interna: ≥,≤ e =.

Para obtener la representación interna de un predicado de autor se aplican las equivalencias descritas en la tabla 11. Obsérvese que un predicado de conocimiento de autor es transformado en uno o varios predicados internos según su tipo.

Tabla 11: De la representación de autor a la representación interna

Predicado de autor (Rk-autor)	Predicado(s) interno(s) equivalente (Rk)
Igual(i_j, valor)	$K(i_j) = valor$
MayorE(i_j, valor)	$K(i_j) \geq valor+1$
MayorNE(i_j, valor)	$K(i_j) \geq valor$
MenorE(i_j, valor)	$K(i_j) \leq valor-1$
MenorNE(i_j, valor)	$K(i_j) \leq valor$
IntervaloE(i_j,val1,val2)	$K(i_j) \leq val2-1$ and $K(i_j) \geq val1+1$
IntervaloNE(i_j,val1,val2)	$K(i_j) \leq val2$ and $K(i_j) \geq val1$

Paso 2. Cuando en Rk-autor(i_j) existe un predicado de autor OR_{SET}, la regla de conocimiento Rk(i_j) es duplicada internamente en *n* reglas de conocimiento $Rk^1(i_j)$, ..., $Rk^n(i_j)$, una por cada predicado dentro del conjunto OR_{SET}.

Todas estas reglas son idénticas a Rk(i_j) salvo que en cada una de ellas se añade la traducción (según la tabla 11) de uno de los *n* predicados del OR_{SET}.

Esto es, dadas las reglas:

Rk-autor(i_j): predicados-autor and $OR_{SET}(i_k,\{predicado^1(i_k),...,predicado^n(i_k)\})$

Rk(i_j): restricciones-internas

, donde restricciones-internas en Rk(i_j) es la traducción según el paso 1 de los predicados-autor distintos de OR_{SET} en Rk-autor(i_j).

La regla Rk(i_j) es trasformada en *n* reglas de conocimiento:

Rk^1(i_j): restricciones-internas and restricciones_internas_i_k^1

. . . . .

Rk^n(i_j): restricciones-internas and restricciones_internas_i_k^n

, donde restricciones_internas_i_k^m es la traducción según el paso 1 del predicado^m(i_k) incluido en el conjunto OR_{SET}.

Paso 3. Para cada Rk^m(i_j) generada en el paso 2 repetir el proceso si existe un predicado OR_{SET} aún no traducido en la Rk-autor.

Paso 4. En cada regla interna Rk^m(i_j) hacer la separación explícita de las restricciones de tipo Restricción^A (≥) y las restricciones de tipo Restricción^I (≤, =), obteniendo así el antecedente de accesibilidad y el antecedente de idoneidad.

Paso 5. Unir el antecedente de accesibilidad y el antecedente de idoneidad mediante el operador lógico *and‴*.

Algoritmo 2. Transformar Rk-autor en Rk

Siguiendo el procedimiento propuesto, la regla de conocimiento definida por el autor en el ejemplo de la figura 2, Rk-autor(I4):

$$\text{IntervaloE}(I1,1,4) \text{ and } OR_{SET}(I2,\{\text{MayorE}(I2,3), \text{MenorNE}(I2,1)\}) \rightarrow \text{Visitar}(I4)$$

Genera en el paso 1, como consecuencia de traducir el predicado de autor IntervaloE(I1,1,4), la regla interna Rk(I4):

$$K(I1) \geq 2 \text{ and } K(I1) \leq 3 \text{ and } \ldots \rightarrow \text{Visitar}(I4)$$

Que durante el paso 2 del algoritmo se desglosa en dos reglas internas, que añaden respectivamente la traducción de los predicados MayorE(I2,3) y MenorNE(I2,1):

$$\text{Rk}^1(I4): K(I1) \geq 2 \text{ and } K(I1) \leq 3 \textit{ and } K(I2) \geq 4 \rightarrow \text{Visitar}(I4)$$

$$\text{Rk}^2(I4): K(I1) \geq 2 \text{ and } K(I1) \leq 3 \textit{ and } K(I2) \leq 1 \rightarrow \text{Visitar}(I4).$$

El paso 3 no es necesario, ya que existe un único OR_{SET}. Los pasos 4 y 5 permiten agrupar los predicados según deban ser evaluados en el antecedente de accesibilidad o en el de idoneidad, quedando las reglas como sigue:

$$\text{Rk}^1(I4): \underbrace{[K(I1)\geq 2 \text{ and } K(I2) \geq 4]}_{\text{Accesibilidad}} \textit{ and‴ } \underbrace{K(I1) \leq 3}_{\text{Idoneidad}} \rightarrow \text{Visitar}(I4)$$

$$\text{Rk}^2(I4): \underbrace{K(I1)\geq 2}_{\text{Accesibilidad}} \textit{ and‴ } \underbrace{[K(I1) \leq 3 \text{ and } K(I2) \leq 1]}_{\text{Idoneidad}} \rightarrow \text{Visitar}(I4)$$

Hagamos un inciso para hacer notar que el uso del predicado OR_{SET} permite al autor expresar en una única regla, Rk-autor^1(I4), que la visita de I4 está permitida pero no es idónea para aquellos usuarios cuyo conocimiento sobre I2 es "medio"(2) o "alto"(3). El autor expresa así que, con un nivel medio-alto, los usuarios conocen de sobra los conceptos básicos explicados en I4 (desconocidos para los que están por debajo de ese nivel), pero aún no son capaces de sacar jugo a los conceptos más elaborados (capacidad que sí tienen los que superan ese nivel).

Por esto, la lectura de I4 está permitida y es idónea para los usuarios con grado de conocimiento "nulo"(0) o "bajo"(1) sobre I2 (regla Rk^2(I4)), y para los que conocen I2 totalmente (regla Rk^1(I4)). Este tipo de prerrequisito pedagógico que discrimina en uno o varios valores intermedios el conocimiento de un ítem requeriría dos reglas de autor si no se proporcionase el predicado OR_{SET}.

Como puede deducirse después del proceso de traducción descrito, al impedir que el autor defina dos predicados sobre el mismo ítem en el cuerpo de una Rk-autor se cumple necesariamente que en ninguna de las Rk generadas como representación interna coexisten dos restricciones sobre el mismo ítem en el mismo antecedente (accesibilidad/idoneidad).

Esto es, sólo hay dos casos en que un predicado de autor genera en su traducción dos o más restricciones internas. El primer caso, corresponde a los predicados de intervalo (IntervaloE e IntervaloNE), cuya traducción genera dos restricciones, pero siempre una de accesibilidad (Restricción^A) y la otra de idoneidad (Restricción^I), por lo que forman parte de antecedentes distintos. El segundo caso, es originado por el predicado OR_{SET}, cuya traducción pueden generar varias restricciones del mismo tipo, pero éstas se separan en reglas diferentes.

Aunque el autor no haya hecho la separación explícita entre accesibilidad e idoneidad debe ser consciente de que las restricciones que exigen un grado de conocimiento mínimo serán utilizadas para prohibir/permitir el acceso al ítem, mientras que las restricciones que exigen un conocimiento máximo o exacto serán utilizadas para informar de la idoneidad de su visita.

6. ACCIONES EVOLUTIVAS DE LAS REGLAS DE CONOCIMIENTO

Para cada regla de autor Rk-autor^i(i_j) asociada al ítem i_j se mantiene internamente un conjunto de reglas {Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j)} equivalente.

- La regla de autor Rk-autor^i(i_j) forma parte del conjunto de reglas de conocimiento Rk-autor(i_j) definidas para i_j.

- Las reglas {Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j)} pertenecen al conjunto de reglas internas asociadas al ítem i_j, esto es Rk(i_j). Estas reglas internas son las que realmente evalúa el sistema para determinar la accesibilidad e idoneidad de la visita del usuario al ítem i_j.

Es necesario proporcionar al autor acciones evolutivas para que pueda crear y modificar las reglas de conocimiento, Rk-autor(EC_A^i), asociadas a una determinada estructura conceptual de aprendizaje. Con relativa frecuencia el autor va a necesitar, por ejemplo,

ampliar los prerrequisitos de conocimiento necesarios para visitar un ítem. Este cambio puede ser requerido por motivos muy diversos: quizá el autor ha modificado su estrategia pedagógica y exige mayor conocimiento previo para reforzar el aprendizaje del ítem o quizá el dominio de información ha aumentado, incluyendo nuevos ítems que contienen información relevante para entender el ítem actual.

Además de ampliar, reducir o modificar los requisitos de conocimiento exigidos en una regla, el autor puede querer añadir nuevas reglas de conocimiento que especifiquen caminos alternativos para poder acceder al contenido de un ítem y/o para que la lectura de su información resulte idónea. De modo inverso, el autor podría desear eliminar alguno de estos caminos que ha dejado de considerar válidos.

Cuando el autor modifica una regla de conocimiento Rk-autor$^{i}(i_j)$ el sistema debe ejecutar los cambios que considere necesarios sobre el conjunto de reglas que internamente representan esa regla de autor. Por este motivo, el sistema mantiene una correspondencia entre la regla de autor Rk-autor$^{i}(i_j)$ y el subconjunto de reglas internas que le corresponden {$Rk^{m}(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$}. Esto permite mantener en todo momento la equivalencia entre el conjunto de reglas de autor definidas para un ítem i_j y su representación interna. Esto es, entre Rk-autor(i_j) y Rk(i_j) y como resultado entre Rk-autor(EC_A^{i}) y Rk(EC_A^{i}).

Cuando el autor modifica una regla de conocimiento existente, se plantean dos alternativas: eliminar su representación interna y generarla de nuevo, o modificar la representación interna en función del cambio realizado. Consideramos que la segunda alternativa, aunque más compleja, es mejor por dos motivos: primero porque, en general, el proceso de traducir una regla de autor conlleva más tiempo que modificar puntualmente su representación interna. Y segundo, porque haciéndolo de esta forma se identifica exactamente el alcance del cambio y se facilita la propagación de éste en los elementos que pueda haber definidos, actualmente o en un futuro, sobre las reglas de conocimiento.

A continuación se enumeran y detallan las acciones evolutivas proporcionadas por el Sistema de Aprendizaje para que el autor pueda definir y modificar las reglas de conocimiento asociadas a cada estructura conceptual de aprendizaje, EC_A^{i}= (EC_M, Rw, MU, EC_P^{j}, $RTnb^{k}$, Ro^{k}, Ru^{i}, Rk^{i}).

Las acciones evolutivas son notadas como ACe[Rk] y numeradas consecutivamente ACe^{1}[Rk], ACe^{2}[Rk], ..., ACe^{n}[Rk]. Cada acción se describe en una tabla, donde se indica:

- los argumentos de la acción evolutiva,
- definición de la modificación realizada,
- precondiciones que deben satisfacerse antes de realizar la acción evolutiva,
- postcondiciones que deben satisfacerse después de ejecutar la acción (si las hay),
- efecto de la acción una vez llevada a cabo,
- salida, esto es, valor devuelto tras la modificación,
- propagación interna realizada como consecuencia del cambio (si es necesario),

- e información sobre quién puede requerir la ejecución de la acción: el autor, el sistema o ambos.

Además se incluyen algunos ejemplos que ilustran la ejecución de la acción evolutiva una vez instanciada, poniendo de manifiesto su desarrollo y utilidad.

6.1 Añadir una regla de conocimiento

Tabla 12: Acciones evolutivas para modificar las reglas de conocimiento –AñadirRk–

ACe^1 [Rk]	**AñadirRk(i_j, Rk-autor^i(i_j)): boolean;**
Argumentos	**i_j** es el identificador del ítem a cuyo conjunto de reglas de conocimiento se va a añadir la nueva regla.
	Rk-autor^i(i_j) es una regla de conocimiento sobre i_j, escrita por el autor usando el lenguaje de autor.
Definición	Esta acción evolutiva añade al conjunto de reglas de conocimiento Rk(i_j) ={ Rk^1(i_j), Rk^2(i_j), ..., Rk^{m-1}(i_j)} asociado a i_j una o varias (*n*) reglas de conocimiento Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) que constituyen la representación interna de la nueva regla Rk-autor^i(i_j). Por supuesto, también se añade la nueva regla de autor Rk-autor^i(i_j) al conjunto de reglas de autor Rk-autor(i_j).
Precondición	El ítem i_j para el que se define la regla de autor y todos los ítems i_k incluidos en el cuerpo de la misma, pertenecen a la estructura conceptual de presentación sobre la que se define la estructura de aprendizaje actual. Esto es, $i_j \in I(EC_P^j)$ y $\forall i_k \in I(EC_P^j)$.
	La regla Rk-autor^i(i_j) cumple las restricciones sintácticas y semánticas especificadas en el lenguaje Rk-autor (secciones 3.1 y 3.2).
	No existe ninguna regla idéntica a Rk-autor^i(i_j) en el conjunto de reglas de autor Rk-autor(i_j). Esto es, Rk-autor^i(i_j) $\notin$ Rk-autor(i_j).
Efecto	El sistema aplica el procedimiento de traducción descrito en la sección 5 y obtiene la representación interna de la nueva regla Rk-autor^i(i_j). Las *n* reglas obtenidas durante el proceso de traducción son notadas respectivamente Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) y se añaden al conjunto de reglas de conocimiento internamente asociadas al ítem i_j. El valor de *m* será igual al número de reglas de conocimiento asociadas a i_j antes de ejecutar la acción evolutiva más una. Esto es, m = |Rk(i_j)| + 1, donde el símbolo | | representa la cardinalidad de un conjunto. Rk(i_j) ={ Rk^1(i_j), Rk^2(i_j), ..., Rk^{m-1}(i_j) } $\Rightarrow$ Rk(i_j) ={ Rk^1(i_j), Rk^2(i_j), ..., Rk^{m-1}(i_j), Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j)}.
	La regla Rk-autor^i(i_j) es guardada en el conjunto de reglas de autor Rk-autor(i_j). Rk-autor(i_j) = { Rk-autor^1(i_j), ..., Rk-autor^{i-1}(i_j)} $\Rightarrow$ Rk-autor(i_j) = { Rk-autor^1(i_j), ..., Rk-autor^{i-1}(i_j), Rk-autor^i(i_j)}.
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando

	que la nueva regla ha sido añadida satisfactoriamente.
Ejecutada_por	Autor.

6.2 Eliminar una regla de conocimiento

Tabla 13: Acciones evolutivas para modificar las reglas de conocimiento –EliminarRk–

ACe^2 [Rk]	**EliminarRk(i_j, Rk-autor^i(i_j)): boolean;**
Argumentos	**i_j** es el identificador del ítem en cuyo conjunto de reglas de conocimiento se desea eliminar la regla.
	Rk-autor^i(i_j) es la regla de conocimiento de autor que se desea eliminar.
Definición	Esta acción evolutiva elimina, del conjunto de reglas de conocimiento Rk(i_j) asociado al ítem i_j, las *n* reglas de conocimiento Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) que constituyen la representación interna de la regla Rk-autor^i(i_j). También se elimina la regla de autor Rk-autor^i(i_j) del conjunto de reglas de autor Rk-autor(i_j).
Precondición	Las reglas Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) pertenecen al conjunto de reglas de conocimiento Rk(i_j) asociado internamente a i_j. Esto es, $\forall s = 0..n \;\; Rk^{m+s}(i_j) \in Rk(i_j)$ O lo que es lo mismo, existe una regla idéntica a Rk-autor^i(i_j) en el conjunto de reglas de autor Rk-autor(i_j).
Efecto	Las reglas Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) son eliminadas del conjunto de reglas de conocimiento asociadas al ítem i_j. Esto es, $\forall s = 0..n \;\; Rk(i_j) = Rk(i_j) - \{Rk^{m+s}(i_j)\}$ Rk(i_j) ={ Rk^1(i_j), ..., Rk^{m-1}(i_j), Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j), Rk^{m+n+1}(i_j), ...} $\Rightarrow$ Rk(i_j) ={ Rk^1(i_j), ..., Rk^{m-1}(i_j), Rk^{m+n+1}(i_j), ... }.
	La regla Rk-autor^i(i_j) es eliminada del conjunto de reglas de autor Rk-autor(i_j). Rk-autor(i_j) = { Rk-autor^1(i_j), ..., Rk-autor^{i-1}(i_j), Rk-autor^i(i_j), Rk-autor^{i+1}(i_j), ...} $\Rightarrow$ Rk-autor(i_j) = { Rk-autor^1(i_j), ..., Rk-autor^{i-1}(i_j), Rk-autor^{i+1}(i_j), ...}.
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que la regla de cocimiento ha sido eliminada satisfactoriamente.
Ejecutada_por	Ambos.

6.3 Eliminar un predicado de una regla de conocimiento

Tabla 14: Acciones evolutivas para modificar las reglas de conocimiento –EliminarPredicadoRk–

ACe^3 [Rk]	**EliminarPredicadoRk(i_j, Rk-autor^i(i_j), i_k): boolean;**
Argumentos	**i_j** es el identificador del ítem cuyo conjunto de reglas de conocimiento Rk(i_j) se desea modificar.
	Rk-autor^i(i_j) determina la regla de conocimiento de la que se quiere eliminar el predicado de autor.
	i_k es el ítem al que se asocia el predicado de conocimiento que se va a eliminar de la regla especificada.
Definición	Esta acción evolutiva elimina la restricción o restricciones de conocimiento impuestas sobre el ítem i_k en el antecedente (de accesibilidad, de idoneidad o de ambos) de las *n* reglas de conocimiento Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) que constituyen la representación interna de la regla Rk-autor^i(i_j). Evidentemente, también se elimina el predicado asociado a i_k en la regla de autor Rk-autor^i(i_j).
Precondición	Las reglas Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) pertenecen al conjunto de reglas de conocimiento Rk(i_j) asociado a i_j. $\forall s = 0..n \ \ Rk^{m+s}(i_j) \in Rk(i_j)$. O lo que es lo mismo, la regla Rk-autor^i(i_j) pertenece al conjunto de reglas de autor definidas para el ítem i_j. Esto es, Rk-autor^i(i_j) $\in$ Rk-autor(i_j).
	Al menos una restricción asociada al ítem i_k aparece en el antecedente (de accesibilidad, de idoneidad o de ambos) de las reglas de conocimiento Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j). O dicho de otro modo, en la regla Rk-autor^i(i_j) existe un predicado definido para el ítem i_k.
Postcondición	No existe en el conjunto de reglas de autor Rk-autor(i_j), ninguna regla idéntica a la regla Rk-autor^i(i_j) después de eliminar en ésta el predicado definido sobre i_k.
Efecto	Paso 1. En cada regla de conocimiento Rk^{m+s}(i_j) con s = 0 ..n : Para cada restricción Restricción^{tipo}(i_k) definida en Rk^{m+s}(i_j) sobre el ítem i_k: eliminarla del antecedente de accesibilidad si *tipo=A* o del antecedente de idoneidad si *tipo=I* . Si después de la eliminación, un antecedente (accesibilidad o idoneidad) queda vacío se sustituye por la fórmula atómica *true*. Paso 2. Si el predicado eliminado es de tipo OR_{SET}, tras el paso 1 existirán varias reglas repetidas en el conjunto de reglas Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j). En este caso, se eliminan todas las reglas idénticas menos una.
	La regla Rk-autor^i(i_j) es modificada eliminando del antecedente el predicado que afecta al ítem i_k.
Propagación Interna	Si el predicado eliminado era el único existente en la regla Rk-autor^i(i_j) y como consecuencia de su eliminación la regla queda vacía, se procede a suprimir dicha regla utilizando para ello la acción evolutiva EliminarRk(i_j, Rk-autor^i(i_j)).

Salida	Si no se satisface alguna de las precondiciones o postcondiciones, la acción evolutiva no tiene efecto y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado se ha eliminado satisfactoriamente.
Ejecutada_por	Ambos.

Dada la regla de conocimiento de autor, Rk-autor^1(I4):

IntervaloE(I1,1,4) and OR_{SET}(I2,{MayorE(I2,3), MenorNE(I2,1)}) $\rightarrow$ Visitar(I4)

Y su representación interna:

Rk^1(I4): [K(I1)$\geq$2 and K(I2) $\geq$4] *and‴* K(I1) $\leq$ 3 $\rightarrow$ Visitar(I4)

Rk^2(I4): K(I1)$\geq$2 *and‴* [K(I1) $\leq$ 3 and K(I2) $\leq$ 1] $\rightarrow$ Visitar(I4)

Si se ejecuta la acción evolutiva *EliminarPredicadoRk(I4, Rk-autor1(I4), I2)* ocurre lo siguiente:

En Rk-autor^1(I4) se elimina el predicado OR_{SET} definido para I2, quedando como sigue:

Rk-autor^1(I4): IntervaloE(I1,1,4) $\rightarrow$ Visitar(I4)

En Rk^1(I4) y Rk^2(I4) se eliminan las restricciones definidas sobre I2 (de accesibilidad en el primer caso y de idoneidad en el segundo).

Rk^1(I4): [K(I1)$\geq$2 and ~~K(I2) $\geq$4~~] *and‴* K(I1) $\leq$ 3 $\rightarrow$ Visitar(I4)

Rk^2(I4): K(I1)$\geq$2 *and‴* [K(I1) $\leq$ 3 and ~~K(I2) $\leq$ 1~~] $\rightarrow$ Visitar(I4)

Tras el cambio, ambas reglas son idénticas. Por lo tanto se deja sólo una de ellas.

Rk^1(I4): K(I1)$\geq$2 *and‴* K(I1) $\leq$ 3 $\rightarrow$ Visitar(I4)

6.4 Añadir un predicado en una regla de conocimiento

Tabla 15: Acciones evolutivas para modificar las reglas de conocimiento –AñadirPredicadoRk–

ACe^4 [Rk]	**AñadirPredicadoRk(i_j, Rk-autor^i(i_j), i_k, predicado(i_k)): boolean;**
Argumentos	**i_j** es el identificador del ítem cuyo conjunto de reglas de conocimiento Rk(i_j) se va a modificar.
	Rk-autor^i(i_j) determina la regla de conocimiento (escrita con semántica de autor) que se desea ampliar.
	i_k es el ítem al que se asocia el predicado de conocimiento que se va a añadir a la regla especificada.

	predicado(i_k) es un predicado lógico de autor que especifica una condición de conocimiento sobre el ítem i_k.
Definición	Esta acción evolutiva añade la restricción o restricciones de conocimiento sobre i_k correspondientes al predicado(i_k) en el antecedente (de accesibilidad, de idoneidad o de ambos) de las *n* reglas de conocimiento $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$ que constituyen la representación interna de la regla Rk-autor$^i(i_j)$. Si predicado(i_k) es de tipo OR_{SET} el resultado de ejecutar esta acción evolutiva aumenta el número de reglas de conocimiento (en su representación interna) asociadas al ítem i_j, ya que cada una de las *n* reglas $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$ se descompone en *p* nuevas reglas, siendo *p* el número de predicados incluidos dentro del conjunto OR_{SET}. Por supuesto, también se añade (mediante el operador *and*) el predicado(i_k) en el cuerpo de la regla de autor Rk-autor$^i(i_j)$.
Precondición	Las reglas $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$ pertenecen al conjunto de reglas de conocimiento asociado al ítem i_j. $\forall s = 0..n \;\; Rk^{m+s}(i_j) \in Rk(i_j)$. O lo que es lo mismo, la regla Rk-autor$^i(i_j)$ pertenece al conjunto de reglas de autor definidas para el ítem i_j. Esto es, Rk-autor$^i(i_j) \in$ Rk-autor(i_j).
	El ítem i_k al que se asocia el nuevo predicado pertenece a la estructura conceptual de presentación sobre la que se define la estructura de aprendizaje actual. Esto es, $i_k \in I(EC_P^j)$.
	No existe ninguna restricción asociada al ítem i_k en el antecedente (de accesibilidad, de idoneidad o de ambos) de las reglas de conocimiento $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$. O de forma equivalente, no existe un predicado definido para el ítem i_k en la regla Rk-autor$^i(i_j)$.
	El predicado(i_k) es correcto tanto sintáctica como semánticamente de acuerdo al lenguaje proporcionado al autor para definir las reglas de conocimiento (secciones 3.1 y 3.2).
Efecto	Siendo $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$ la representación interna de la regla de conocimiento Rk-autor$^i(i_j)$:
	- Si predicado(i_k) es de tipo $OR_{SET}(i_k,\{predicado^1(i_k), ..., predicado^p(i_k)\})$: Paso 1. Para cada predicado predicado$^r(i_k)$ dentro del OR_{SET} (r =1,...,p): 1.1 Obtener la restricción o combinación de restricciones equivalente en representación interna (véase la tabla 11). predicado$^r(i_k) \equiv$ Restricción$^A(i_k)$ y/o Restricción$^I(i_k)$. 1.2. Para cada regla de conocimiento $Rk^{m+s}(i_j)$ (s=0,...,n): Crear una nueva regla idéntica a ella y añadirle con un *and* la restricción Restricción$^A(i_k)$ en el antecedente de accesibilidad y/o la restricción Restricción$^I(i_k)$ en el antecedente de idoneidad. Paso 2. Añadir al conjunto Rk(i_j) las nuevas reglas creadas durante el paso 1 y eliminar el conjunto original de reglas $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$.

	- En otro caso: Paso 1. Obtener la traducción en restricciones internas del predicado(i_k) según la tabla 11. predicado(i_k) ≡ Restricción^A(i_k) y/o Restricción^I(i_k). Paso 2. Para cada regla de conocimiento Rk^{m+s}(i_j) (s=0,1,...,n) : Modificar la regla añadiéndole, mediante el operador lógico *and*, la restricción Restricción^A(i_k) al antecedente de accesibilidad y/o la restricción Restricción^I(i_k) al antecedente de idoneidad.
	En cualquier caso, la regla Rk-autor^i(i_j) se modifica incluyendo en su cuerpo el nuevo predicado sobre i_k.
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado se ha añadido satisfactoriamente.
Ejecutada_por	Autor.

6.5 Modificar un predicado distinto de OR_{SET} en una regla de conocimiento

Tabla 16: Acciones evolutivas para modificar las reglas de conocimiento –ModificarPredicadoRk–

ACe^5 [Rk]	**ModificarPredicadoRk(i_j, Rk-autor^i(i_j), i_k, val1, val2): boolean;**
Argumentos	**i_j** es el identificador del ítem cuyo conjunto de reglas de conocimiento Rk(i_j) se va a modificar.
	Rk-autor^i(i_j) identifica la regla de conocimiento de autor que se desea modificar.
	i_k es el ítem al que se asocia en la regla Rk-autor^i(i_j) el predicado de conocimiento que se va a modificar. Siendo predicado(i_k) un predicado de cualquier tipo excepto OR_{SET}.
	val1 es el nuevo valor para la primera variable del predicado definido en Rk-autor^i(i_j) sobre i_k. Si predicado(i_k) tiene dos variables (como es el caso de IntervaloE, IntervaloNE), el autor puede dar valor -1 a este argumento para indicar que no desea modificar el valor de la primera variable.
	val2 es el nuevo valor para la segunda variable del predicado. Será -1 si se trata de un predicado de una sola variable (Igual, MayorE, MayorNE, MenorE, MenorNE) o si el autor no desea modificar la segunda variable de un predicado de tipo Intervalo.
Definición	Esta acción evolutiva modifica convenientemente la restricción o restricciones de conocimiento sobre i_k correspondientes al predicado(i_k) en el antecedente (de accesibilidad, de idoneidad o de ambos) de las *n* reglas de conocimiento Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^{m+n}(i_j) que constituyen la representación interna de la regla de autor Rk-autor^i(i_j). Por descontado, en la regla de autor Rk-autor^i(i_j) se modifica, según lo requerido, la(las) variable (s) del predicado definido sobre i_k.

Precondición	Las reglas $Rk^{m}(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$ pertenecen al conjunto de reglas de conocimiento internamente asociadas al ítem i_j. $\forall s = 0..n \;\; Rk^{m+s}(i_j) \in Rk(i_j)$. Es decir, la regla Rk-autor$^{i}(i_j)$ pertenece al conjunto de reglas de autor definidas para el ítem i_j. Rk-autor$^{i}(i_j) \in$ Rk-autor(i_j).
	Una restricción asociada al ítem i_k aparece en el antecedente (de accesibilidad, de idoneidad o de ambos) de las reglas de conocimiento $Rk^{m}(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$. Dicho de otro modo, en la regla de autor Rk-autor$^{i}(i_j)$ existe un predicado asociado al ítem i_k.
	Si el predicado definido sobre el ítem i_k en Rk-autor$^{i}(i_j)$ es de tipo MayorE, MayorNE, MenorE, MenorNE o Igual, *val1* debe ser distinto de -1 y *val2* igual a -1. Esto es, val1 ≠ -1 and val2==-1. Si por el contrario, predicado(i_k) es de tipo IntervaloE o IntervaloNE, basta con que *val1* o *val2* sea distinto de -1, pero no pueden serlo ambos. Esto es, val1 ≠ -1 or val2 ≠ -1.
	Los nuevos valores propuestos para *val1* y/o *val2* deben cumplir las restricciones semánticas impuestas para el tipo de predicado definido sobre i_k (véase la tabla 4).
Efecto	Paso 1. Modificar la instanciación de las variables del predicado(i_k) en la regla de autor Rk-autor$^{i}(i_j)$. Si predicado(i_k) tiene sólo una variable, ésta se instancia con val1. Si predicado(i_k) tiene dos variables: La primera variable se instancia con el nuevo valor val1 si val1 ≠ -1 o se deja como está si val1==-1. Del mismo modo, la segunda variable se instancia con val2 si val2 ≠ -1 o se queda con el valor actual si val2==-1.
	Paso 2. Obtener la restricción o combinación de restricciones equivalente en representación interna al predicado(i_k) instanciado con los nuevos valores. Véase la tabla 11. predicado(i_k) ≡ K(i_k)≥ val_nuevo1 y/o K(i_k) ≤ val_nuevo2. Paso 3. Para cada regla de conocimiento $Rk^{m+s}(i_j)$ (s=0,...,n): Si aparece una restricción sobre i_k en el antecedente de accesibilidad sustituir el valor exigido por val_nuevo1. Asimismo, si existe una restricción definida sobre i_k en el antecedente de idoneidad sustituir el valor requerido por val_nuevo2.
Salida	Si no se satisface alguna de las precondiciones exigidas, la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado se ha modificado satisfactoriamente.
Ejecutada_por	Autor.

Dada la regla de conocimiento de autor, Rk-autor^{1}(I4):

$$\text{IntervaloE}(I1,1,4) \text{ and } OR_{SET}(I2,\{\text{MayorE}(I2,3), \text{MenorNE}(I2,1)\}) \rightarrow \text{Visitar}(I4)$$

Y su representación interna:

Rk^1(I4): [K(I1)≥2 and K(I2) ≥4] *and‴* K(I1) ≤ 3 → Visitar(I4)

Rk^2(I4): K(I1)≥2 *and‴* [K(I1) ≤ 3 and K(I2) ≤ 1] → Visitar(I4)

Si se ejecuta la acción evolutiva *ModificarPredicadoRk(I4, Rk-autor^1(I4), I1, -1, 3)* ocurre lo siguiente:

En Rk-autor^1(I4) se modifica el valor del segundo argumento del predicado Intervalo definido sobre I1, quedando como sigue (cambio en negrita):

IntervaloE(I1,1,**3**) and OR_{SET}(I2,{MayorE(I2,3), MenorNE(I2,1)})→ Visitar(I4)

Con objeto de hacer evolucionar la representación interna, se traduce el predicado modificado, Intervalo(I1,1,3) ≡ K(I1) ≥ 2 y K(I1) ≤ 2.

En Rk^1(I4) y Rk^2(I4) se buscan las restricciones definidas sobre I1 y si el valor exigido no coincide con el de la traducción se modifica. Esto ocurre en el antecedente de idoneidad de ambas reglas (cambios en negrita).

Rk^1(I4): [K(I1)≥2 and K(I2) ≥4] *and‴* K(I1) ≤ **2** → Visitar(I4)

Rk^2(I4): K(I1)≥2 *and‴* [K(I1) ≤ **2** and K(I2) ≤ 1] → Visitar(I4)

6.6 Modificar un predicado OR_{SET} en una regla de conocimiento

Tabla 17: Acciones evolutivas para modificar las reglas de conocimiento –ModificarOR_{SET}PredicadoRk–

ACe[6] [Rk]	**ModificarOR_{SET}PredicadoRk(i_j, Rk-autor^i(i_j), i_k, predicado^t(i_k), v1, v2): boolean;**
Argumentos	**i_j** es el identificador del ítem cuyo conjunto de reglas de conocimiento se va a modificar.
	Rk-autor^i(i_j) determina la regla de conocimiento de autor que se desea modificar.
	i_k es el ítem al que se asocia en la regla Rk-autor^i(i_j) el predicado que se quiere modificar. Siendo este predicado, predicado(i_k), un predicado de tipo OR_{SET}(i_k, {predicado^1(i_k), …, predicado^p(i_k)}).
	predicado^t(i_k) representa al predicado que se desea añadir, eliminar o modificar en el conjunto OR_{SET} definido sobre el ítem i_k. Obviamente afecta al ítem i_k y puede ser de cualquier tipo excepto, de nuevo, OR_{SET}.
	v1 es el nuevo valor para la primera variable del predicado^t(i_k). El autor da valor -1 a este argumento si no se desea cambiar la instanciación de dicha variable.
	v2 es el nuevo valor para la segunda variable del predicado^t(i_k). Será –1 si se trata de un predicado de una sola variable o si no se desea modificar.

Definición	Esta acción evolutiva modifica el predicado OR_{SET} definido sobre el ítem i_k en la regla de autor Rk-autor$^i(i_j)$. La modificación realizada depende del valor asignado por el autor a los argumentos de la acción evolutiva: Si ambos, v1y v2 son -1 (v1==-1 and v2==-1): Se elimina el predicado$^t(i_k)$ del conjunto OR_{SET}. En otro caso (v1 ≠ -1 or v2 ≠ -1): Si predicado$^t(i_k)$ pertenece al conjunto OR_{SET}: Se modifica la instanciación de predicado$^t(i_k)$ con los nuevos valores v1 y/o v2. Si predicado$^t(i_k)$ no pertenece al conjunto OR_{SET}: Se añade el nuevo predicado, predicado$^t(i_k)$, al conjunto. En este caso, v1 y v2 indican la instanciación del nuevo predicado. Como consecuencia, esta acción evolutiva realiza diferentes modificaciones sobre el conjunto de las *n* reglas Rk$^m(i_j)$, Rk$^{m+1}(i_j)$, ..., Rk$^{m+n}(i_j)$ que constituyen la representación interna de la regla Rk-autor$^i(i_j)$. Así dependiendo del tipo de modificación solicitada por el autor, se eliminan reglas del conjunto Rk(i_j), se añaden nuevas reglas o se modifican determinadas restricciones en las reglas existentes.
Precondición	Las reglas Rk$^m(i_j)$, Rk$^{m+1}(i_j)$, ..., Rk$^{m+n}(i_j)$ pertenecen al conjunto de reglas de conocimiento Rk(i_j) asociado al ítem i_j. $\forall s = 0..n \;\; Rk^{m+s}(i_j) \in Rk(i_j)$. O dicho de otro modo, la regla Rk-autor$^i(i_j)$ pertenece al conjunto de reglas definidas por el autor para i_j. Rk-autor$^i(i_j)$ ∈ Rk-autor(i_j).
	Si v1 y v2 son igual a -1, el predicado a eliminar debe pertenecer al conjunto OR_{SET} definido para i_k. Es decir, predicado$^t(i_k)$ ∈ {predicado$^1(i_k)$, …, predicado$^p(i_k)$}. O lo que es lo mismo, $\exists_1^p j \,/\, j == t$ Además, el conjunto OR_{SET} debe contener al menos tres predicados, para que tras la eliminación de predicado$^t(i_k)$ el conjunto contenga como mínimo dos predicados. Es decir, se debe satisfacer *p*>2.
	Si v1 o v2 son distintos de -1 y predicado$^t(i_k)$ pertenece al conjunto OR_{SET}, esto es, predicado$^t(i_k)$ ∈ {predicado$^1(i_k)$, …, predicado$^p(i_k)$}: El predicado predicado$^t(i_k)$ será modificado con los nuevos valores v1 y/o v2. Por lo tanto, si v1 ≠ -1, v1 debe cumplir las restricciones semánticas de la primera variable de este tipo de predicado. Y asimismo, si v2 ≠ -1, v2 debe cumplir las restricciones de la segunda variable (véase la tabla 4).
	Si v1 o v2 son distintos de -1 y predicado$^t(i_k)$ no aparece en el conjunto OR_{SET}, esto es, predicado$^t(i_k)$ ∉ {predicado$^1(i_k)$, …, predicado$^p(i_k)$}: El predicado predicado$^t(i_k)$ será añadido al conjunto OR_{SET}, y por lo tanto debe ser un predicado sintáctica y semánticamente correcto (véase sección 3). El tipo del predicado puede ser cualquiera excepto OR_{SET}.
Efecto	Siendo Rk$^m(i_j)$, Rk$^{m+1}(i_j)$, ..., Rk$^{m+n}(i_j)$ la representación interna de la regla de conocimiento Rk-autor$^i(i_j)$.

	- Si v1==v2==-1: Paso 1. Obtener la restricción o restricciones internas equivalentes al predicado predicado$^t(i_k)$ según lo especificado en la tabla 11 y de acuerdo a su instanciación actual. predicado$^t(i_k) \equiv K(i_k) \geq$ v_a1 y/o $K(i_k) \leq$ v_a2 Paso 2. Eliminar del subconjunto de reglas internas {$Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$} todas aquellas en cuyo cuerpo aparezca la traducción exacta de predicado$^t(i_k)$. Esto es, la restricción $K(i_k) \geq$ v_a1 en el antecedente de accesibilidad y/o la restricción $K(i_k) \leq$ v_a2 en el antecedente de idoneidad.
	- Si v1 ≠ -1 o v2 ≠ -1 y predicado$^t(i_k) \notin$ {predicado$^1(i_k)$, …, predicado$^p(i_k)$}: Paso 1. Obtener la restricción o restricciones internas equivalentes al predicado que se va a añadir: predicado$^t(i_k)$. (véase la tabla 11). predicado$^t(i_k) \equiv K(i_k) \geq$ v_a1 y/o $K(i_k) \leq$ v_a2 Paso 2. Hacer una copia en R de las reglas $Rk^{m+s}(i_j)$ con s=0,...,n. Y eliminarlas de $Rk(i_j)$. 2.1 Eliminar del cuerpo de cada regla perteneciente a R todas las restricciones impuestas sobre el ítem i_k. 2.2 Como consecuencia del paso 2.1 existirán en R reglas idénticas, siendo necesario eliminar las reglas repetidas (dejar sólo una). 2.3 A cada regla en R le añado (*and*) las restricciones sobre i_k obtenidas en el paso 1. Esto es, $K(i_k) \geq$ v_a1 en el antecedente de accesibilidad y/o $K(i_k) \leq$ v_a2 en el antecedente de idoneidad. 2.4 Añado las reglas de R al subconjunto de reglas {$Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$} que representan internamente Rk-autor$^i(i_j)$.
	- Si v1 ≠ -1 o v2 ≠ -1 y predicado$^t(i_k) \in$ {predicado$^1(i_k)$, …, predicado$^p(i_k)$}: Paso 1. Obtener la traducción interna del predicado predicado$^t(i_k)$ de acuerdo a su instanciación actual (tabla 11). predicado$^t(i_k) \equiv K(i_k) \geq$ v_a1 y/o $K(i_k) \leq$ v_a2 Paso 2. Obtener la traducción interna del predicado$^t(i_k)$, pero ahora instanciado con los valores especificados por los argumentos de la acción evolutiva v1 y v2. (véase la tabla 11). predicado$^t(i_k)' \equiv K(i_k) \geq$ v_n1 y/o $K(i_k) \leq$ v_n2 Paso 3. En el cuerpo de aquellas reglas del subconjunto {$Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$} donde aparece exactamente la traducción obtenida en el paso 1 se actualizan los valores de las restricciones (accesibilidad y/o idoneidad) con los obtenidos en la traducción del paso 2, esto es v_n1 y/o v_n2.
	En el cuerpo de la regla Rk-autor$^i(i_j)$ se modifica el predicado sobre i_k: $OR_{SET}(i_k$, {predicado$^1(i_k)$, ..., predicado$^p(i_k)$}) eliminando, añadiendo o modificando el predicado$^t(i_k)$ de acuerdo a los argumentos de la acción evolutiva.
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado OR_{SET} se ha modificado satisfactoriamente.

Ejecutada_por	Autor.

Dada la regla de conocimiento de autor, Rk-autor^1(I5):

IntervaloNE(I3,1,3) and OR_{SET}(I6,{MenorNE(I6,3), Igual(I6,4), MayorE(I6,2)}) → Visitar(I5)

Y su representación interna:

Rk^1(I5): K(I3)≥1 *and‴* [K(I3) ≤ 3 and K(I6) ≤ 3] → Visitar(I5)

Rk^2(I5): K(I3)≥1 *and‴* [K(I3) ≤ 3 and K(I6) = 4] → Visitar(I5)

Rk^3(I5): [K(I3)≥1 and K(I6) ≥3] *and‴* K(I3) ≤ 3 → Visitar(I5)

Si se ejecuta la acción evolutiva *ModificarOR_{SET}PredicadoRk(I5, Rk-autor^1(I5), I6, Igual(I6,4), -1, -1)* ocurre lo siguiente:

En Rk-autor^1(I5) se elimina el predicado Igual(I6,4) del conjunto OR_{SET} definido sobre I6, quedando la regla como sigue:

IntervaloNE(I3,1,3) and OR_{SET}(I6,{MenorNE(I6,3), MayorE(I6,2)}) → Visitar(I5)

Con objeto de hacer evolucionar la representación interna, se traduce el predicado eliminado, Igual(I6, 4) ≡ K(I6) = 4. Y se elimina la regla que contiene la traducción.

Rk^1(I5): K(I3)≥1 *and‴* [K(I3) ≤ 3 and K(I6) ≤ 3] → Visitar(I5)

~~Rk^2(I5): K(I3)≥1 *and‴* [K(I3) ≤ 3 and **K(I6) = 4**] → Visitar(I5)~~

Rk^3(I5): [K(I3)≥1 and K(I6) ≥3] *and‴* K(I3) ≤ 3 → Visitar(I5)

6.7 Conjunto de acciones evolutivas

A continuación, todas las acciones evolutivas aplicables sobre las reglas de conocimiento se resumen en la tabla 18, contemplando únicamente: su nombre, utilidad e implicaciones y responsable de su ejecución.

Tabla 18: Acciones evolutivas para modificar las reglas de conocimiento

ACe[Rk]	Utilidad/Implicaciones	
AñadirRk	Se utiliza para añadir una nueva regla al conjunto de reglas de conocimiento de la EC_A. Esto implica la adicción de una regla de autor al conjunto Rk-autor(i_j) y una o varias reglas internas al conjunto Rk(i_j).	Autor

AñadirPredicadoRk	Se utiliza para añadir un predicado en el cuerpo de una regla de conocimiento previamente definida en la EC_A. Esto implica la adicción del nuevo predicado en el cuerpo de la regla de autor Rk-autor^i(i_j). Asimismo se debe agregar una restricción, en el antecedente de accesibilidad y/o en el antecedente de idoneidad, de una o varias reglas internas. Si el predicado es de tipo OR_{SET} se crean nuevas reglas internas en Rk(i_j).	Sistema y Autor
ModificarPredicadoRk	Se utiliza para cambiar los valores que restringen el grado de conocimiento en un predicado existente en una regla de conocimiento de la EC_A. No es aplicable para predicados de tipo OR_{SET}. Esto implica la modificación de la instanciación del predicado en la regla de autor Rk-autor(i_j). Y, como consecuencia, la actualización de la(s) restricción(es) que constituyen su traducción interna en las reglas asociadas internamente a esa regla de autor { $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$ }.	
ModificarOR_{SET}PredicadoRk	Se utiliza para modificar un predicado OR_{SET} existente en una regla de conocimiento Rk-autor^i(i_j), en alguna de estas tres formas: añadiendo un nuevo predicado al conjunto, eliminando un predicado del conjunto o cambiando la instanciación de un predicado del conjunto. Añadir un predicado dentro del OR_{SET} implica la generación de nuevas reglas internas en Rk(i_j). Eliminar un predicado dentro del OR_{SET} supone la eliminación de algunas de estas reglas internas. Por último, modificar un predicado dentro del OR_{SET} implica actualizar convenientemente las restricciones equivalentes en una o varias reglas internas.	
EliminarRk	Se utiliza para eliminar una regla de conocimiento del conjunto de reglas de conocimiento Rk-autor definidas en la EC_A. Esto implica, además, la eliminación del subconjunto de reglas mantenidas internamente en Rk(i_j) como representación de la regla de autor Rk-autor^i(i_j) eliminada.	
EliminarPredicadoRk	Se utiliza para eliminar un predicado existente en el cuerpo de una determinada regla de conocimiento. Esto implica, la eliminación del predicado en la regla de autor Rk-autor^i(i_j) y la eliminación de su traducción en una o varias reglas internas: $Rk^m(i_j)$, $Rk^{m+1}(i_j)$, ..., $Rk^{m+n}(i_j)$. Si el predicado es de tipo OR_{SET} como consecuencia se eliminan una o varias de estas reglas en Rk(i_j).	

CAPÍTULO 9

Reglas de Actualización

Resumen

En este capítulo se define el formalismo utilizado para actualizar de modo automático el conocimiento que el usuario adquiere a medida que inspecciona los ítems de información. En primer lugar, se establecen las circunstancias en que la visita a un ítem debe producir un incremento de conocimiento en el usuario. Se describe el procedimiento utilizado por el sistema para generar un conjunto de reglas de actualización por defecto, y cómo el autor puede modificar éstas. Se especifica la sintaxis y la semántica del lenguaje proporcionado al autor para establecer las actualizaciones deseadas. Y, por último, se fijan y justifican las restricciones que el autor debe satisfacer durante la modificación de una regla de actualización, detallando las acciones evolutivas proporcionadas para ello.

Tabla de contenidos

Reglas de Actualización

1. INTRODUCCIÓN

Las reglas de actualización averiguan el conocimiento del usuario sobre los ítems del sistema hipermedia basándose en las visitas que éste realiza durante su proceso de navegación en dicho sistema. Expresado de otro modo, es el mecanismo utilizado para calcular de forma automática cómo cambia el estado de conocimiento del usuario a medida que éste consulta los ítems de información ofrecidos.

> **Def 9.1 [Regla de actualización 1]** La regla de actualización asociada al ítem i_j determina, después de una visita a i_j, cuál es el incremento de conocimiento del usuario respecto a i_j, y posiblemente a otros ítems relacionados.

La ejecución de la regla de actualización tiene sentido, claro está, sólo cuando el usuario está preparado para asimilar el contenido del ítem al que accede, esto es, si su estado de conocimiento antes de la visita satisface las restricciones de accesibilidad impuestas en alguna regla de conocimiento asociada a dicho ítem (capítulo 8). Obviamente, si no existen restricciones para acceder al ítem, su visita siempre desencadena la ejecución de la regla de actualización.

> **Def 9.2 [Regla de actualización 2]** Existe un conjunto de reglas de actualización para cada estructura conceptual de aprendizaje EC_A^i, al que notamos de forma abreviada $Ru(EC_A^i)$. La cardinalidad del conjunto $Ru(EC_A^i)$ coincide con el número de ítems mostrados en la estructura conceptual de presentación sobre la que se define la EC_A^i. Concretamente existe una regla de actualización para cada ítem i_j incluido en dicha presentación, a la cuál notamos $Ru(i_j)$ de forma unívoca dentro de la EC_A^i.

2. ¿CUÁNDO SE DEBE EJECUTAR UNA REGLA DE ACTUALIZACIÓN?

Como se ha mencionado anteriormente, la regla de actualización $Ru(i_j)$ asociada a un ítem i_j es ejecutada cada vez que el usuario visita dicho ítem cumpliendo las restricciones de accesibilidad exigidas para ello. De esta forma, después de que el usuario acceda al contenido del ítem i_j, el sistema actualiza su estado de conocimiento con el nuevo conocimiento adquirido.

Sin embargo, somos conscientes de que el hecho de que un usuario visite un ítem no asegura que haya leído completa y suficientemente la información que contiene, ni por lo tanto que haya comprendido su significado. En consecuencia, se debería exigir para la ejecución de la regla de actualización asociada a un ítem, el cumplimiento de determinadas condiciones más restrictivas que la mera inspección del mismo. Dos alternativas posibles serían:

a) Tiempo mínimo de lectura:

Esta solución consiste en establecer un tiempo mínimo que un ítem debe permanecer abierto para considerar su visita como válida. Este tiempo puede ser distinto para cada ítem y podría equivaler al tiempo necesario para leer con suficiente profundidad y

detenimiento la información contenida en el ítem. Obviamente, a esta alternativa se le siguen escapando multitud de situaciones, ya que no se asegura que durante el tiempo de exhibición del ítem el usuario esté dedicado a su lectura.

b) Evaluación de la asimilación de contenidos:

Esta solución implica asociar a cada ítem un cuestionario, que el usuario debe rellenar si desea dejar constancia de la visita. El sistema corrige de forma automática las respuestas al cuestionario, obteniendo una puntuación para el usuario. Dicha puntuación deberá superar un valor mínimo para considerar válida la visita y que se ejecute su regla de actualización.

Esta alternativa plantea una serie de problemas o dificultades. Por ejemplo, la corrección automática del cuestionario es trivial cuando se trata de preguntas tipo *test*, pero para plantear problemas más complejos es necesario utilizar técnicas de inteligencia artificial [Murria, 99], que permitan identificar los errores cometidos por el usuario, averiguar sus posibles causas, etc.

Otras cuestiones serían: ¿No es excesivo que cada vez que el usuario visite un ítem resuelva el cuestionario asociado?, ¿qué ocurre si el usuario suspende un cuestionario que ha aprobado en anteriores ocasiones?,... Obviamente la respuesta a estas preguntas y a otras similares debe ser competencia del autor. Del mismo modo, la elaboración de los cuestionarios constituye otra importante tarea de autor.

A partir de lo anterior se deduce que, si bien esta alternativa garantiza la comprensión básica del ítem por parte del usuario (dejando de lado los problemas de suplantación de identidad), también requiere un esfuerzo, nada despreciable, por parte del autor y los usuarios. Este trabajo adicional derivado del diseño, elaboración y resolución de cuestionarios puede estar justificado en sistemas hipermedia educacionales, pero creemos que es exagerado imponerlo en todas las posibles aplicaciones hipermedia del modelo propuesto.

Puesto que la decisión de cuándo ejecutar una regla de actualización parece tener una fuerte dependencia de la aplicación concreta del sistema hipermedia, dejamos por ahora esta decisión, que deberá retomarse durante la implementación de cada sistema. Hasta entonces, consideramos por simplicidad que se exige como única condición la visita a un ítem (por supuesto accesible) para que su regla de actualización sea ejecutada.

3. DEFINICIÓN DE UNA REGLA DE ACTUALIZACIÓN

3.1 Estructura general de una Ru

Las reglas de actualización son reglas de la forma *si-entonces*. Por lo tanto, la estructura de una regla de actualización $Ru(i_j)$ puede dividirse en dos partes bien diferenciadas: *cabeza* y *cuerpo*. Véase en la ecuación 1 la sintaxis de una regla de actualización para i_j.

<regla_actualización> ::= Ru ‘(’ <i_j> ‘)’ ‘:’ Si <cabeza> entonces <cuerpo> ‘;’ (1)

En la **<cabeza>** se especifica la condición que debe cumplirse para que sea ejecutado el cuerpo. En este caso, la condición exigida es que el usuario visite el ítem i_j para el que

se define la regla, estando además, en posesión de un estado de conocimiento que satisfaga las restricciones impuestas para el acceso a i_j en $Rk(i_j)$. Esta condición se representa con un predicado lógico **Visitado(i_j)** que se hace cierto cada vez que el usuario accede al contenido del ítem i_j cumpliendo las restricciones de accesibilidad necesarias. Véase la ecuación 2.

<cabeza> ::= Visitado '(' <i_j> ')' (2)

El **<cuerpo>** contiene uno o más predicados de actualización que especifican cómo se actualiza el conocimiento del usuario sobre el ítem visitado y posiblemente otros ítems relacionados con éste. Cada predicado del cuerpo debe ser instanciado sobre un ítem diferente, y es representado de forma general como **Actualización(i_k)**. Así pues, no pueden existir dos predicados de actualización asociados al mismo ítem en el cuerpo de una regla de actualización, situación que obviamente no tendría sentido.

Siempre el primer predicado de actualización del cuerpo de una regla $Ru(i_j)$ está asociado al ítem visitado, esto es al ítem de la cabeza i_j, y se ejecuta en primer lugar. Además se debe garantizar que el predicado de actualización establecido para la cabeza permite que el usuario alcance un grado de conocimiento "total" sobre este ítem mediante la visita repetida al mismo y el conocimiento previo necesario para que se ejecute la regla. La ecuación 3 muestra la definición del cuerpo de la regla de actualización $Ru(i_j)$.

<cuerpo> ::= <Actualización(i_j)> {,<Actualización(i_k)>} (3)

De acuerdo a las especificaciones BNF de las ecuaciones 1, 2 y 3, la estructura general de una regla de actualización es la que se muestra en la ecuación 4 donde se resalta en negrita la parte obligatoria.

$Ru(i_j)$: Si Visitado(i_j) entonces Actualización(i_j), Actualización(i_k), ..., Actualización(i_m)**;** (4)

3.2 Predicados de actualización

El orden de evaluación de los predicados de actualización existentes en el cuerpo de una regla $Ru(i_j)$ es indiferente, salvo para la actualización del ítem de la cabeza que debe ser ejecutada siempre antes que el resto. Esto se debe a que, como se explica más adelante, algunos predicados del cuerpo pueden hacer referencia al grado de conocimiento actualizado sobre el ítem visitado.

Se necesitan predicados de actualización de diferente tipo, ya que, después de la lectura de un ítem, el conocimiento del usuario sobre un determinado conjunto de ítems va a cambiar, pero seguramente no de igual forma, ni en el mismo grado. Los predicados de actualización que proporciona el Sistema de Aprendizaje son los que aparecen en el último nivel del diagrama mostrado en la figura 1.

En el diagrama los predicados aparecen clasificados según tres características. Esto se debe a que la actualización del grado de conocimiento sobre un ítem responde siempre a tres cuestiones: ¿qué tipo de cantidad? (1), ¿en relación a qué? (2) y ¿durante cuánto tiempo? (3). Cada una de estas cuestiones admite dos alternativas posibles:

1. **Tipo de cantidad**: *incremental* (se aplican incrementos o decrementos para calcular el nuevo grado de conocimiento sobre el ítem) o *fija* (el conocimiento sobre el ítem es igualado a un grado de conocimiento fijo),

2. **En relación a qué**: *absoluta* (se usa una etiqueta semántica) o *relativa* (se usa el grado de conocimiento que el usuario posee sobre el ítem visitado),

3. **Durante cuando tiempo**: *1ª-vez* (la actualización sólo se ejecuta la primera vez que el ítem de la cabeza es visitado) o *cada-vez* (la actualización se evalúa cada vez que el ítem cabeza se visita).

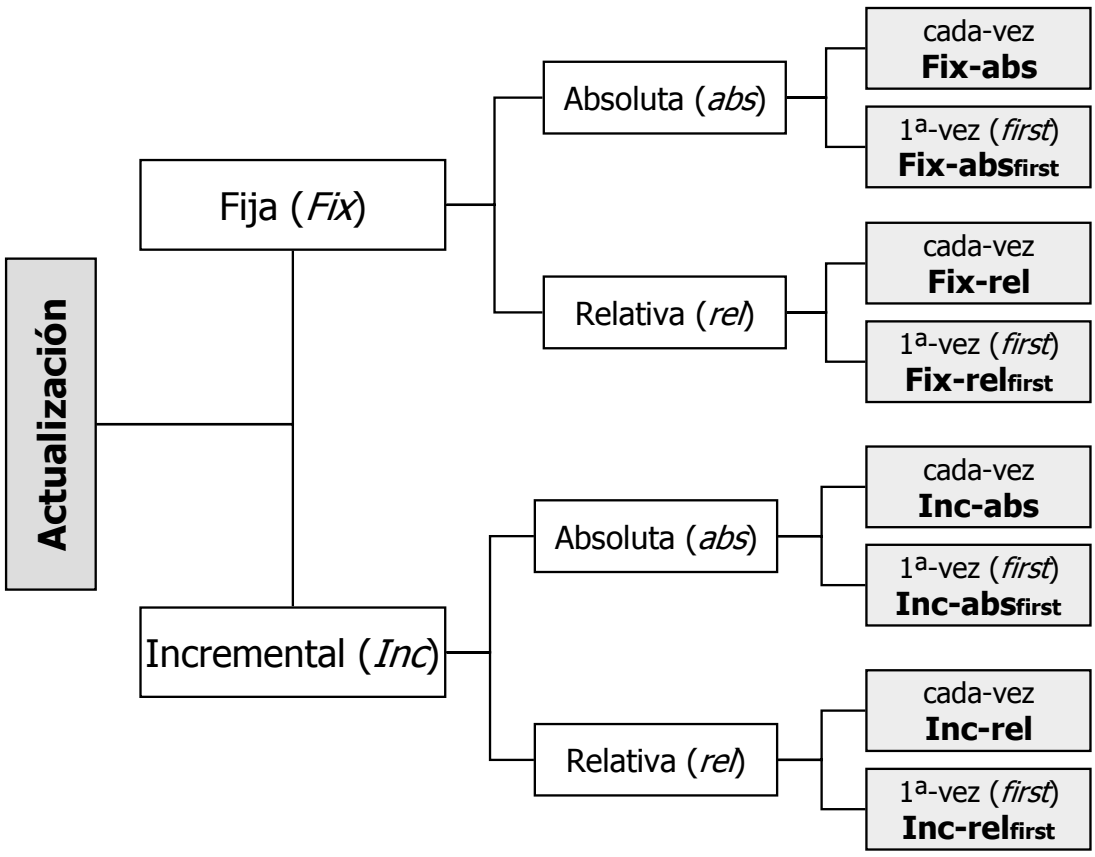


Figura 1: Predicados de actualización

Siguiendo los cuadros del diagrama que aparecen en el camino que une el cuadro inicial (Actualización) con el cuadro que contiene el predicado deseado, podemos ver las características de dicho predicado. Por ejemplo, el predicado *Fix-abs* supone una actualización fija y absoluta cada vez que se visita el ítem de la cabeza. Mientras que el predicado $Inc\text{-}rel_{first}$ realiza una actualización incremental y relativa sólo después de que el usuario visite por primera vez el ítem de la cabeza.

4. Sintaxis y Semántica de cada Predicado de Actualización

La sintaxis de cada uno de los predicados mostrados en el diagrama de la figura 1, se especifica en la tabla 1 usando notación BNF.

Tabla 1: Sintaxis de los predicados de actualización

<Actualización(i_j)> ::= <Fix-abs(i_j)> | <Fix-abs$_{first}$(i_j)> | <Fix-rel(i_j)> | <Fix-rel$_{first}$(i_j)> | <Inc-abs(i_j)> | <Inc-abs$_{first}$(i_j)> | <Inc-rel(i_j)> | <Inc-rel$_{first}$(i_j)>
<Fix-abs(i_j)> ::= Fix-abs '(' <identificador_ítem>, <etiqueta_semántica> ')'
<Fix-abs$_{first}$(i_j)> ::= Fix-abs$_{first}$ '(' <identificador_ítem>, <etiqueta_semántica> ')'
<Fix-rel(i_j)> ::= Fix-rel '(' <identificador_ítem> ')'
<Fix-rel(i_j) $_{first}$> ::= Fix-rel$_{first}$ '(' <identificador_ítem> ')'
<Inc-abs(i_j)> ::= Inc-abs '(' <identificador_ítem>, <incremento> ')'
<Inc-abs$_{first}$(i_j)> ::= Inc-abs$_{first}$ '(' <identificador_ítem>, <incremento> ')'
<Inc-rel(i_j)> ::= Inc-rel '(' <identificador_ítem>, (<incremento>|<decremento>) ')'
<Inc-rel(i_j) $_{first}$> ::= Inc-rel$_{first}$ '(' <identificador_ítem>, (<incremento>|<decremento>) ')'
<identificador_ítem> ::= "El identificador de algún ítem existente en la EC_A"
<etiqueta_semántica> ::= "nulo" | "bajo" | "medio" | "alto" | "total"
<incremento> ::= [1-4]
<decremento> ::= – <incremento>

A continuación se describe cada uno de los predicados de actualización en detalle, indicando, además de sus características, el número y tipo de sus argumentos, las condiciones que deben cumplirse para la ejecución del predicado y la actualización producida tras esta ejecución. También se muestran algunos ejemplos de uso que contribuyen a facilitar su comprensión.

Sólo se describen los predicados *cada-vez*, ya que su significado es estrictamente el mismo que el de los predicados *1ª-vez*, salvo que estos últimos sólo son ejecutados si es la primera vez que el usuario accede al ítem de la cabeza en la estructura de navegación actual.

Notamos que a diferencia de las reglas de conocimiento donde el autor utilizaba el valor numérico de las etiquetas semánticas en lugar de su lexema, en los predicados de actualización el autor trabaja con el propio lexema ("nulo", "bajo", "medio", "alto" y "total"). Esto permite que le sea más fácil distinguir el valor numérico correspondiente a un incremento o decremento de un valor de conocimiento. Internamente el sistema trabaja con valores numéricos[1] porque esto facilita las comprobaciones y actualizaciones necesarias. En cualquier caso, siempre que se utiliza una etiqueta

[1] Para especificar los predicados se asume que existen cinco etiquetas semánticas. En consecuencia, el valor numérico máximo para una etiqueta es 4. Si hubiese un número distinto de etiquetas bastaría reemplazar el valor 4 por el nuevo número de etiquetas menos uno.

semántica se puede utilizar su valor numérico, puesto que son totalmente equivalentes para el sistema. Véase la definición de etiqueta semántica en el capítulo 8.

Tabla 2: Predicados de actualización –Fix-abs–

Ru(i_j) : Si Visitado(i_j) entonces ..., **Fix-abs(i_k, et_{SEM}^S)**, ...;		
Características	Actualización que **fija** el conocimiento sobre el ítem i_k a un valor **absoluto**.	
Argumentos	**i_k** es el ítem cuyo grado de conocimiento se va a actualizar. **et_{SEM}^S** es la etiqueta semántica implicada en la actualización.	
Condición	La actualización sólo se realiza si el valor numérico de et_{SEM}^S es mayor que el grado de conocimiento que actualmente posee el usuario sobre el ítem i_k. Es decir, debe satisfacerse: valor-numérico(et_{SEM}^S) > K(i_k). Por lo tanto, sea cual sea la et_{SEM}^S implicada, la actualización no se realiza si el usuario posee ya el grado de conocimiento máximo sobre el ítem i_k, esto es K(i_k)=4("total"). Dicho de otro modo, debe satisfacerse que: K(i_k) < 4.	
Actualización	Después de visitar el ítem accesible i_j el nuevo grado de conocimiento que el usuario posee sobre el ítem i_k, notado K'(i_k), está representado por la etiqueta semántica et_{SEM}^S. Esto es, K'(i_k) = valor-numérico(et_{SEM}^S).	
Ejemplo	Si Visitado(i_j) entonces ..., Fix-abs(i_k, "medio"), ...;	
	Antes: K(i_k) = 3 ("alto")	Después: K'(i_k) = 3 ("alto")
	Antes: K(i_k) = 1 ("bajo")	Después: K'(i_k) = 2 ("medio")

Tabla 3: Predicados de actualización –Fix-rel–

Ru(i_j) : Si Visitado(i_j) entonces Actualización(i_j),..., **Fix-rel(i_k)**, ...;		
Características	Actualización que **fija** el conocimiento sobre i_k de forma **relativa** al grado de conocimiento alcanzado sobre i_j tras la visita y actualización de éste.	
Argumentos	**i_k** es el ítem cuyo grado de conocimiento se va a actualizar.	
Condición	La actualización sólo se realiza si el grado de conocimiento que alcanza el usuario sobre i_j, después de ejecutar la actualización especificada en el predicado Actualización(i_j), es mayor que el grado de conocimiento actual del usuario sobre i_k. Es decir, debe satisfacerse: K(i_j)' > K(i_k), siendo K'(i_j) el resultado de aplicar el predicado Actualización(i_j) sobre K(i_j). La actualización sólo se considera si el conocimiento del usuario sobre i_k no es aún "total", esto es si K(i_k) < 4.	
Actualización	Después de visitar el ítem accesible i_j el grado de conocimiento que el usuario posee sobre el ítem i_k es igual que el grado de conocimiento que posee sobre i_j tras ejecutar su actualización. Esto es K'(i_k) = K'(i_j).	
Ejemplo	Si Visitado(i_j) entonces Fix-abs(i_j,"medio"),..., **Fix-rel(i_k)**,...;	
	Antes: K(i_k) = 3	Después:K'(i_j) = 2, K'(i_k) = 3
	Antes: K(i_k) = 1	Después: K'(i_j) = 2, K'(i_k) = 2

Tabla 4: Predicados de actualización –Inc-abs–

Ru(i_j) : Si Visitado(i_j) entonces ..., **Inc-abs(i_k, num)**, ...;		
Características	Actualización que **incrementa** el conocimiento sobre i_k en un número **absoluto** de grados.	
Argumentos	**i_k** es el ítem cuyo grado de conocimiento se va a actualizar. **num** es un número natural mayor que 0 y menor o igual que 4, el cuál será utilizado en la actualización. num $\in$ N y 0 < num $\leq$ 4.	
Condición	La actualización se realiza si el grado de conocimiento del usuario sobre el ítem i_k es menor que "total". Esto es, K(i_k) < 4.	
Actualización	Después de visitar el ítem i_j el conocimiento que el usuario posee actualmente sobre el ítem i_k se incrementa en el número de grados que se especifica en el argumento num. El resultado del incremento es normalizado a 4, esto es, si el resultado de la suma es mayor que 4 se rebaja a 4. K'(i_k) = mínimo {K(i_k) + num, 4}.	
Ejemplo	Si Visitado(i_j) entonces..., **Inc-abs(i_k, 2)**, ...;	
	Antes: K(i_k) = 3	Después: K'(i_k) = 4
	Antes: K(i_k) = 1	Después: K'(i_k) = 3

Tabla 5: Predicados de actualización –Inc-rel–

Ru(i_j) : Si Visitado(i_j) entonces Actualización(i_j),..., **Inc-rel(i_k, num)**, ...;	
Características	Actualización que obtiene el nuevo grado de conocimiento sobre i_k **incrementando o decrementando** el conocimiento **relativo** al ítem de la cabeza, i_j, después de ser visitado y actualizado.
Argumentos	**i_k** es el ítem cuyo grado de conocimiento se va a actualizar. **num** es un número entero distinto de 0, mayor que -4 y menor o igual que 4, el cuál será utilizado en la actualización. num $\in$ Z, num $\neq$ 0 y -4 < num $\leq$ 4.
Condición	Para que se ejecute la actualización sobre i_k el grado de conocimiento del usuario sobre dicho ítem debe ser menor que el resultado de aplicar el incremento/decremento de grados especificado en el argumento num sobre el conocimiento i_j tras ejecutar el predicado Actualización(i_j). Esto es: K(i_k) < (K'(i_j) + num), donde K'(i_j) es el conocimiento actualizado de i_j. Por supuesto, la actualización ni siquiera se considera si el conocimiento del usuario sobre i_k es ya "total" antes de la visita a i_j. Por lo tanto, se debe satisfacer: K(i_k) < 4.
Actualización	Después de visitar el ítem accesible i_j el conocimiento que el usuario posee sobre el ítem i_k se iguala al grado de conocimiento que logra sobre i_j después de ejecutar su actualización, Actualización(i_j), incrementado (si num es positivo) o decrementado (si num es negativo) en num grados. Esto es: K'(i_k) = mínimo {K'(i_j) + num, 4}

Ejemplo	Si Visitado(i_j) entonces Inc-abs(i_j,2), ..., **Inc-rel(i_k, 1)**, ...;	
	Antes: K(i_j) = 0, K(i_k) = 0	Después: K'(i_j) = 2, K'(i_k) = 3
	Si Visitado(i_j) entonces Inc-abs(i_j,2), ..., **Inc-rel(i_k, -1)**, ...;	
	Antes: K(i_j) = 0, K(i_k) = 0	Después: K'(i_j) = 2, K'(i_k) = 1

A partir de la especificación semántica de los predicados de actualización, se puede ver de forma más clara porqué la actualización sobre el ítem de la cabeza debe ser ejecutada en primer lugar. Como se adelantaba en la sección anterior, el resultado de este predicado es utilizado en los predicados de actualización relativa existentes en el cuerpo de la regla. De este modo estos predicados trabajan con el grado de conocimiento del usuario sobre el ítem de la cabeza, ya actualizado.

5. ¿QUIÉN DEFINE LAS REGLAS DE ACTUALIZACIÓN?

5.1 Generación de reglas de actualización por defecto

El autor tiene capacidad para definir las reglas de actualización que considere adecuadas, usando para ello los predicados proporcionados con la sintaxis y semántica descrita anteriormente.

Para cada estructura conceptual de aprendizaje, $EC_A^i = (EC_M, Rw, MU, EC_P^k, RTnb^j, Ro^j, Ru^i, Rk^i)$, debe existir un conjunto de reglas de actualización $Ru(EC_A^i)$, donde cada ítem i_j existente en el dominio de información de EC_P^k tiene asociada una regla $Ru(i_j)$.

Para asistir al autor en esta tarea, el sistema genera por defecto un conjunto de reglas de actualización asociado a cada estructura conceptual de presentación EC_P^k. Este conjunto constituye inicialmente el conjunto de reglas de actualización $Ru(EC_A^i)$ asociado a todas las estructuras de aprendizaje EC_A^i definidas a partir de esa presentación.

Concretamente, para cada ítem i_j mostrado en la presentación EC_P^k, el sistema crea un regla $Ru(i_j)$ en cuyo cuerpo existe un único predicado de actualización de tipo fijo y absoluto. Este predicado fija en "total" el grado de conocimiento sobre el ítem después de ser visitado. La ecuación 5 muestra la sintaxis de una regla generada por defecto:

$$Ru(i_j): \text{Si Visitado}(i_j) \text{ entonces Fix-abs}(i_j, \text{"total"}); \quad (5)$$

5.2 Modificación de las reglas de actualización

El autor puede modificar la regla de actualización asociada a un ítem i_j en una estructura de aprendizaje EC_A^i siempre que lo crea oportuno. Por ejemplo, puede completar la regla por defecto de un ítem, añadiendo, en su cuerpo, predicados de actualización sobre otros ítems relacionados con el ítem cabeza, de tal forma que al conocer más acerca de éste último se aprenda, también, algo más en relación con aquellos.

Este enfoque de adquisición de conocimiento sigue la teoría constructiva del conocimiento, de acuerdo a la cual, existen múltiples relaciones entre los ítems de

información. Lo que implica que al aprender algo acerca de uno de ellos se aumente el conocimiento sobre varios ítems relacionados (véase el capítulo 1).

De este modo, el cambio sufrido por el estado de conocimiento del usuario tras visitar un ítem no tiene porque afectar únicamente al ítem visitado. Esto es, dado el estado de conocimiento inicial:

$$estado_K = \{<i_1, K(i_1)>, <i_2, K(i_2)>, ...,<i_j, K(i_j)>,...,<i_k, K(i_k)>,..., <i_n, K(i_n)>, ...\}$$

tras ejecutar una regla de actualización $Ru(i_j)$ es posible obtener el nuevo estado:

$$estado_K' = \{<i_1, K(i_1)>, <i_2, K(i_2)>, ..., <i_j, K'(i_j)>,..., <i_k, K'(i_k)>,..., <i_n, K'(i_n)>, ...\}$$

que difiere del anterior en el grado de conocimiento sobre i_j, pero además en el de otros ítems como i_k o i_n. Véase la definición de estado de conocimiento en el capítulo 8.

5.2.1 Restricciones durante la modificación de una Ru

A la hora de hacer evolucionar una regla de actualización, esto es, añadir, eliminar o modificar los predicados de su cuerpo, el autor debe ajustarse a una serie de restricciones, de cuyo obligado cumplimiento se ocupa el sistema. Estas restricciones se enumeran y describen a continuación, y posteriormente en la sección 6 se incluyen (y formalizan) como precondiciones de las acciones evolutivas correspondientes.

Puede observarse que, la mayoría de las restricciones tienen como objetivo asegurar que el usuario alcance un conocimiento "total" sobre un ítem si visita repetidamente el contenido de dicho ítem, claro está, en posesión del conocimiento previo requerido para su comprensión (es decir, satisfaciendo las restricciones de accesibilidad asociadas).

Esta característica, además de ser conveniente y lógica a nivel conceptual[2], garantiza que la navegación sobre una EC_A logre conocimiento "total" sobre todos los ítems mostrados, siempre que esté asegurada la ausencia de ítems inalcanzables. Esto es, debe garantizarse que en algún momento todo ítem se vuelve accesible para el usuario de ahí en adelante. La forma concreta en que se comprueba y garantíza esta condición se establece en el siguiente capítulo (capítulo 10, Consistencia $Rk \cup Ru$).

Las restricciones son las siguientes:

a) Tal y como se define en la ecuación 3, en el cuerpo de una regla de actualización siempre debe existir un predicado de actualización sobre el ítem de la cabeza. Por lo tanto, no se permite al autor eliminar dicho predicado, tan sólo modificarlo y de acuerdo a las restricciones expresadas desde b) hasta e), ambas inclusive.

b) El predicado de actualización sobre el ítem de la cabeza no puede ser relativo, ya que este tipo de predicado hace referencia al conocimiento del ítem cabeza después de ser actualizado y dicha actualización está, precisamente, en proceso.

[2] Sería frustrante para un usuario que posee, de acuerdo a lo especificado por el autor, el conocimiento necesario para comprender un ítem, que por muchas veces que estudie su contenido no consiga conocimiento "total" sobre él.

c) Puesto que la ejecución repetida de una regla de actualización debe lograr un conocimiento "total" sobre el ítem de la cabeza, si el predicado de actualización sobre dicho ítem es fijo tiene que implicar necesariamente la etiqueta semántica "total". Es decir debe ser un predicado del tipo Fix-abs(i_j, "total") donde i_j es el ítem de la cabeza.

d) Por el mismo motivo que en el apartado c), si el predicado de actualización sobre el ítem de la cabeza es incremental tiene que ser obligatoriamente de tipo *cada-vez* para que el incremento se aplique en todas las visitas hasta alcanzar un conocimiento "total". En este caso el predicado de la cabeza debe ser de la forma: Inc-abs(i_j, num).

e) Debido a la misma causa, aunque el predicado de actualización sobre el ítem de la cabeza sea fijo tampoco puede ser del tipo *1ª-vez*. Para comprender mejor el porqué de esta restricción, es interesante hacer explícita la diferencia existente entre los predicados Fix-abs y Fix-abs$_{first}$.

Supongamos que la regla de actualización asociada a un ítem i_j es la siguiente:

Ru(i_j): Si Visitado(i_j) entonces Inc-abs(i_j,2);

un usuario con conocimiento "nulo" acerca de i_j necesita dos visitas para conocerlo totalmente y una para alcanzar un grado de conocimiento "medio". Supongamos que este segundo es el caso de nuestro usuario.

Si ahora el autor modifica la regla Ru(i_j), sustituyendo el predicado Inc-abs(i_j, 2) por Fix-abs$_{first}$(i_j, "total"), la nueva regla:

Ru(i_j): Si Visit(i_j) entonces Fix-abs$_{first}$(i_j,"total");

no actualiza el conocimiento sobre i_j para aquellos usuarios que hayan visitado previamente i_j. Por esto, nuestro usuario no puede obtener, por muchas veces que visite i_j un conocimiento mayor que el alcanzado antes de la modificación de la regla Ru(i_j), esto es "medio".

Si en lugar del predicado Fix-abs$_{first}$ el autor modifica Ru(i_j) utilizando el predicado Fix-abs no tiene cabida el anterior problema, ya que la ejecución de este predicado se realiza siempre que el usuario visite el ítem i_j, independientemente de si lo ha visitado antes o no. Por lo tanto, nuestro usuario tras su segunda visita a i_j obtendría conocimiento "total" sobre él.

---

f) En el cuerpo de una regla de actualización no pueden existir dos predicados de actualización sobre el mismo ítem, ya que en este caso no habría forma de decidir cual de los dos prevalece.

Como se expuso anteriormente, las restricciones a), b), c), d) y e) garantizan que si el usuario visita *n* veces (*n*>= 1 y *n* finito) un determinado ítem acabará adquiriendo un conocimiento "total" sobre éste. Por su parte, el conjunto de restricciones b), c), d) y e) implican que los únicos predicados de actualización permitidos sobre el ítem de la cabeza son: Fix-abs(i_j, "total") e Inc-abs(i_j, num).

5.2.2 Interfaz de autor para la modificación de Ru

Para modificar las reglas de actualización, el sistema debe ofrecer al autor una interfaz gráfica, que le facilite esta tarea, de forma que pueda ignorar por completo la sintaxis de la regla de actualización y los predicados que la componen. Además, es interesante que la propia interfaz impida al autor la violación de las restricciones. Por ejemplo, permitiendo una única actualización sobre cada ítem, o restringiendo los predicados de actualización viables para el ítem de la cabeza.

En esta sección describimos una interfaz que podría ser utilizada para modificar una regla de actualización Ru(I1). En la interfaz propuesta (figura 2) los ítems de la regla son resaltados sobre una red semántica que representa la estructura conceptual donde se define la regla.

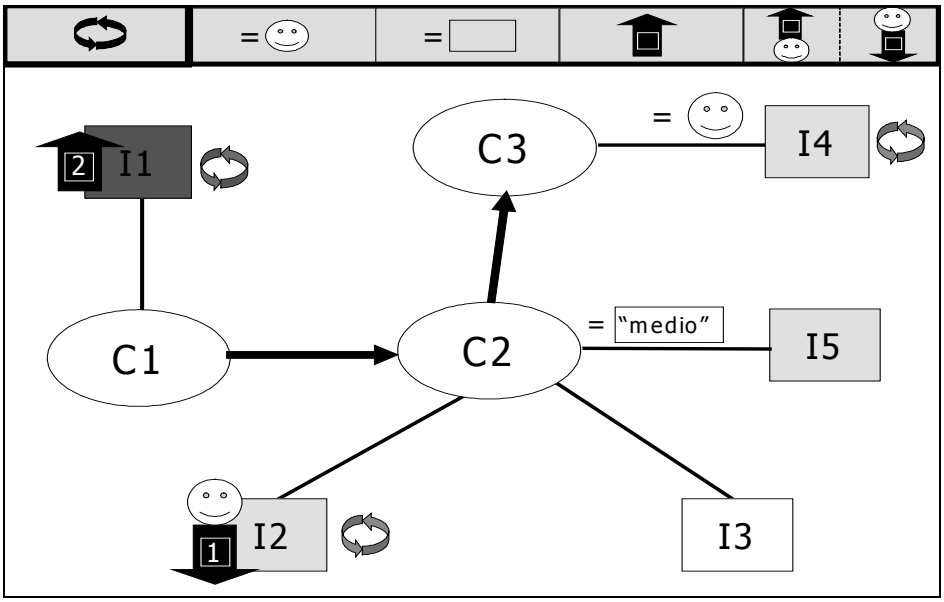


Figura 2: Regla de actualización asociada al ítem I1

- El ítem cabeza, I1 en el ejemplo, aparece resaltado en color oscuro.

- Cada vez que el autor selecciona un ítem para definir sobre él una actualización, este es marcado en color claro, indicando así que forma parte del cuerpo de la regla (I2, I4 e I5 en el ejemplo).

La actualización sobre cada ítem es expresada mediante símbolos gráficos colocados junto a éste. La interpretación de estos símbolos es la siguiente:

- Si a la derecha del ítem aparece un símbolo compuesto por dos flechas circulares significa que la actualización es de tipo “cada-vez”. La ausencia de dicho símbolo indica que la actualización es de tipo “1ª-vez”.

- Una actualización “fija” es siempre encabezada por el símbolo de igualdad (=). Si la actualización fija es “absoluta” tras éste debe aparecer el grado de conocimiento al que se fija, expresado mediante una etiqueta semántica. Si la actualización es “relativa” un icono de cara sonriente aparece junto al símbolo =, indicando que la actualización es “relativa” y, por lo tanto, se fija el conocimiento del ítem marcado al grado de conocimiento alcanzado sobre el ítem cabeza.

- La presencia de una flecha gruesa de color negro establece una actualización "incremental". Si la flecha apunta hacia arriba supone un "incremento" (aplicable en actualizaciones "absolutas" y "relativas") y si apunta hacia abajo implica un "decremento" (sólo aplicable en actualizaciones "relativas"). Si la actualización es "relativa", la flecha va acompañada de la cara sonriente. Y en cualquier caso, dentro de la flecha aparece escrito el número de grados en que se incrementa o decrementa.

El autor puede definir la actualización sobre un ítem seleccionado previamente, arrastrando hasta él el icono que desee desde la barra de herramientas superior (véase la figura 2). El primer icono sirve para establecer si la actualización del ítem debe ser considerada sólo tras la primera visita del usuario al ítem cabeza o después de cada visita. Los últimos cinco iconos, simbolizan en este orden, tipos de actualización: fija y relativa, fija y absoluta, incremental y absoluta, incremental y relativa con incremento, e incremental y relativa con decremento.

Por ejemplo, el cuerpo de la regla definida para el ítem I1 en la figura 2 tiene cuatro predicados de actualización, uno para el propio I1, y otros tres para I2, I4 e I5 respectivamente.

Usando los predicados de actualización descritos en la sección 4, la regla asociada a I1 es la siguiente:

Ru(I1): Si Visitado(I1) entonces Inc-abs(I1, 2), Inc-rel(I2, -1), Fix-rel(I4), Fix-abs$_{first}$(I5, "medio");

Supongamos un usuario que tiene el estado de conocimiento representado parcialmente en la tabla 6 y no ha visitado aún el ítem I1.

Tabla 6: Estado de conocimiento parcial antes de visitar I1

K(I1)	K(I2)	K(I3)	K(I4)	K(I5)	...
0	0	1	3	1	

Si dicho usuario accede al contenido del ítem I1 satisfaciendo las restricciones de accesibilidad requeridas en sus reglas de conocimiento, la visita provocará la ejecución de la regla definida en la figura 2 y, como consecuencia, un cambio en su estado de conocimiento que queda como se muestra en la tabla 7.

Tabla 7: Estado de conocimiento parcial después de visitar I1

K(I1)	K(I2)	K(I3)	K(I4)	K(I5)	...
2	1	1	3	2	

Adviértase que con esta interfaz, el autor puede seleccionar cualquier ítem que desee actualizar en el cuerpo de la regla sin más que hacer *clic* sobre él. Una vez seleccionado, puede definir su predicado de actualización de forma bastante intuitiva. Además, eliminar el predicado asociado a un ítem en el cuerpo de la regla puede ser tan fácil como deseleccionar dicho ítem. También se permite modificar los valores del predicado de actualización asociado a un ítem, e incluso cambiar el tipo de predicado.

Por supuesto, no está permitido cualquier cambio. Durante el uso de la interfaz, es necesario controlar el cumplimiento de las restricciones explicadas en la sección 5.2.1. De esto se encargan las acciones evolutivas que son las responsables a nivel interno de realizar los cambios especificados por el autor a través de la interfaz gráfica. Así, un cambio no es realizado si no satisface las precondiciones exigidas en la acción evolutiva correspondiente. En tal caso, el autor obtiene un mensaje en la interfaz que le informa de la imposibilidad de realizar ese cambio.

Algunas de las cosas que se controlan son las siguientes:

- No se pueden introducir dos predicados de actualización sobre el mismo ítem, esto es, el autor debe elegir uno sólo de los cinco últimos iconos de la barra de herramientas (figura 2), pudiendo ir cada uno de éstos acompañado o no del primer icono.

- Una regla no está completa hasta que el ítem marcado en oscuro, el ítem cabeza, tenga un predicado válido. Este ítem no se puede deseleccionar, ni se permite eliminar el símbolo de actualización "cada-vez" asociado a él. En caso de desear cambiar su actualización, de los últimos cinco iconos sólo se permite usar el tercero y el cuarto, ya que el resto representan predicados no válidos para la cabeza. Además, en caso de optar por la actualización fija (icono tercero) el único grado de conocimiento permitido es "total".

6. ACCIONES EVOLUTIVAS DE LAS REGLAS DE ACTUALIZACIÓN

A continuación se describen las acciones evolutivas que permiten crear, eliminar y modificar las reglas de actualización asociadas a una estructura conceptual de aprendizaje, $EC_A^i = (EC_M, Rw, MU, EC_P^k, RTnb^j, Ro^j, Ru^i, Rk^i)$. De nuevo, para cada acción, se especifica: argumentos, definición, precondiciones, efecto, salida, propagación interna y ejecutada_por (autor, sistema o ambos).

6.1 Crear una regla de actualización

Tabla 8: Acciones evolutivas para modificar las reglas de actualización –CrearRu–

ACe[1] [Ru]	**CrearRu(i_j)**
Argumentos	**i_j** es el identificador del ítem al que se desea asociar la nueva regla de actualización.
Definición	Esta acción evolutiva crea en $Ru(EC_A^i)$ una nueva regla de actualización $Ru(i_j)$ que será ejecutada tras la visita del ítem i_j siempre que el usuario reúna el conocimiento previo requerido para dicho acceso en las reglas de conocimiento $Rk(i_j) \in Rk(EC_A^i)$. La regla $Ru(i_j)$ es generada tal y como se especifica en la ecuación 5. Esto es, con un único predicado de actualización en el cuerpo, que fija a "total" el conocimiento sobre el ítem i_j.
Precondición	El ítem i_j pertenece al dominio de información de la presentación EC_P^k incluida en la estructura de aprendizaje EC_A^i. Esto es, $i_j \in I(EC_P^k)$.

	No existe ninguna regla $Ru(i_j)$ asociada al ítem i_j en el conjunto de reglas de actualización de la estructura de aprendizaje actual. Esto es, $Ru(i_j) \notin Ru(EC_A^i)$
Efecto	Se crea la regla por defecto $Ru(i_j)$: Si Visitado(i_j) entonces Fix-abs(i_j, "total"); y se añade al conjunto de reglas de la estructura de aprendizaje actual. Esto es, $Ru'(EC_A^i) = Ru(EC_A^i) \cup \{Ru(i_j)\}$.
Salida	Esta acción evolutiva es ejecutada por el sistema sólo cuando se muestra un nuevo ítem en la presentación EC_P^k sobre la que se define la EC_A^i actual. Por lo tanto, se garantiza la satisfacción de las precondiciones y no es necesaria ninguna salida explícita (véase la propagación del cambio en el capítulo 13).
Ejecutada_por	Sistema.

6.2 Eliminar una regla de actualización

Tabla 9: Acciones evolutivas para modificar las reglas de actualización –EliminarRu–

ACe² [Ru]	**EliminarRu(i_j)**
Argumentos	**i_j** es el identificador del ítem cuya regla de actualización se desea eliminar.
Definición	Esta acción evolutiva elimina la regla de actualización $Ru(i_j)$ del conjunto de reglas de actualización, $Ru(EC_A^i)$, de la estructura de aprendizaje actual.
Precondición	Existe una regla $Ru(i_j)$ asociada al ítem i_j en el conjunto de reglas de actualización de la estructura de aprendizaje actual. Esto es, $Ru(i_j) \in Ru(EC_A^i)$.
	No existe en la presentación EC_P^k asociada a la estructura de aprendizaje EC_A^i ningún ítem identificado como i_j. Esto es, $i_j \notin I(EC_P^k)$.
Efecto	Se elimina la regla de actualización del conjunto $Ru(EC_A^i)$, reduciendo su cardinalidad en un elemento. Esto es, $Ru'(EC_A^i) = Ru(EC_A^i) - \{Ru(i_j)\}$.
Salida	Esta acción evolutiva es ejecutada por el sistema sólo cuando se oculta un ítem de la presentación EC_P^k sobre la que se define la EC_A^i, por lo tanto se garantiza la satisfacción de las precondiciones exigidas y no es necesaria ninguna salida.
Ejecutada_por	Sistema.

6.3 Añadir un predicado de actualización en una regla

Tabla 10: Acciones evolutivas para modificar las reglas de actualización –AñadirPredicadoRu–

ACe^3 [Ru]	AñadirPredicadoRu(i_j, i_k, predicado(i_k)): boolean;
Argumentos	**i_j** es el identificador del ítem en cuya regla de actualización se quiere añadir el predicado. **i_k** es el identificador del ítem sobre el que se define el nuevo predicado. **predicado(i_k)** es un predicado de actualización definido sobre i_k.
Definición	Esta acción evolutiva añade el predicado de actualización predicado(i_k) en el cuerpo de la regla Ru(i_j). La regla resultante es idéntica a la inicial salvo en el nuevo predicado, el cuál hará que tras visitar i_j se actualice, si cabe, el conocimiento sobre i_k. Esto permite al autor, poner de manifiesto y hacer efectiva, la relación de adquisición de conocimiento existente entre la información proporcionada por ambos ítems, i_k e i_j.
Precondición	Existe una regla Ru(i_j) asociada al ítem i_j en el conjunto de reglas de actualización de la estructura de aprendizaje actual. Esto es, Ru(i_j) $\in$ Ru(EC_A^i).
	El ítem i_k pertenece a la presentación EC_P^k incluida en la estructura de aprendizaje actual. Esto es, $i_k \in I(EC_P^k)$.
	No existe actualmente en el cuerpo de la regla Ru(i_j) ningún predicado de actualización que afecte al mismo ítem, i_k, que el nuevo predicado. Esto es, siendo Ru(i_j): Si Visitado(i_j) entonces <cuerpo>; se cumple que <actualización(i_k)> $\notin$ <cuerpo>.
	La instanciación de predicado(i_k) es correcta desde un punto de vista sintáctico y semántico (véase la sección 4).
Efecto	El nuevo predicado se añade al final del cuerpo de la regla Ru(i_j). Ru(i_j): Si Visitado(i_j) entonces cuerpo; $\Rightarrow$ Ru'(i_j): Si Visitado(i_j) entonces cuerpo, **predicado(i_k)**;
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado se ha añadido satisfactoriamente.
Ejecutada_por	Autor.

6.4 Eliminar un predicado de actualización en una regla

Tabla 11: Acciones evolutivas para modificar las reglas de actualización –EliminarPredicadoRu–

ACe^4 [Ru]	EliminarPredicadoRu(i_j, i_k): boolean;
Argumentos	**i_j** es el identificador del ítem en cuya regla de actualización se quiere eliminar un predicado existente. **i_k** es el identificador del ítem al que afecta el predicado de actualización que se desea eliminar.

Definición	Esta acción evolutiva elimina el predicado de actualización asociado al ítem i_k en el cuerpo de la regla Ru(i_j).
Precondición	Existe una regla Ru(i_j) asociada al ítem i_j en el conjunto de reglas de actualización de la estructura de aprendizaje actual. Esto es, Ru(i_j) $\in$ Ru(EC_A^i).
	Existe en el cuerpo de la regla Ru(i_j) un predicado que actualiza i_k. Esto es, siendo Ru(i_j): Si Visitado(i_j) entonces <cuerpo>; se cumple que <actualización(i_k)> $\in$ <cuerpo>.
	El ítem i_k es distinto del ítem i_j cabeza de la regla. Esto es: $i_k \neq i_j$. Esta precondición es necesaria para impedir que se elimine la actualización del ítem visitado.
Efecto	En el cuerpo de la regla Ru(i_j) el predicado de actualización sobre i_k es eliminado. Ru(i_j): Si Visitado(i_j) entonces <cuerpo'>, **predicado(i_k)**, <cuerpo''>; $\Rightarrow$ Ru'(i_j): Si Visitado(i_j) entonces <cuerpo'>, <cuerpo''>; , donde <cuerpo'> y <cuerpo''> representan el conjunto de predicados de actualización situados a la izquierda y a la derecha respectivamente del predicado(i_k). <cuerpo''> puede ser vacío, pero <cuerpo'> tiene como mínimo un predicado (el de la cabeza).
Salida	Si no se satisface alguna de las precondiciones especificadas, la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado ha sido eliminado.
Ejecutada_por	Ambos (autor y sistema).

6.5 Modificar un predicado de actualización en una regla

Tabla 12: Acciones evolutivas para modificar las reglas de actualización –ModificarPredicadoRu–

ACe[5] [Ru]	**ModificarPredicadoRu(i_j, i_k, predicado'(i_k)): boolean;**
Argumentos	**i_j** es el identificador del ítem en cuya regla de actualización se quiere modificar un predicado existente. **i_k** es el identificador del ítem sobre el que se define el predicado de actualización que se desea modificar. **predicado'(i_k)** es el predicado de actualización sobre i_k ya modificado.
Definición	Esta acción evolutiva sustituye el predicado de actualización del cuerpo de la regla Ru(i_j) que afecta al ítem i_k por otro predicado, predicado'(i_k) definido sobre ese mismo ítem. La modificación puede consistir en cambiar simplemente los valores de actualización del predicado o definir un predicado de actualización de distinto tipo. Si el predicado modificado es el que afecta al ítem cabeza, i_j, sólo se permiten las modificaciones que cumplan las restricciones establecidas en la sección 5.2.1.

Precondición	Existe una regla Ru(i_j) asociada al ítem i_j en el conjunto de reglas de actualización de la estructura de aprendizaje actual. Esto es, Ru(i_j) $\in$ Ru(EC_A^i).
	Existe, actualmente, en el cuerpo de la regla Ru(i_j) un predicado que actualiza i_k. Esto es, siendo Ru(i_j): Si Visitado(i_j) entonces <cuerpo>; se cumple que <actualización(i_k)> $\in$ <cuerpo>.
	La sintaxis del predicado modificado, predicado'(i_k) es correcta de acuerdo a lo especificado en la sección 4.
	Si el ítem i_k es el ítem cabeza de la regla, esto es $i_k = i_j$, el predicado modificado, predicado'(i_k), debe ser válido según las restricciones establecidas para la cabeza (sección 5.2.1): predicado'(i_k) = Fix-abs(i_j, "total") o predicado'(i_k) = Inc-abs(i_j, *num*) con $1 \leq num \leq 4$.
Efecto	En el cuerpo de la regla Ru(i_j) el predicado modificado, predicado'(i_k), sustituye al antiguo predicado predicado(i_k). Ru(i_j): Si Visitado(i_j) entonces <cuerpo'> predicado(i_k) <cuerpo''>; $\Rightarrow$ Ru'(i_j): Si Visitado(i_j) entonces <cuerpo'> **predicado'(i_k)** <cuerpo''>; , donde <cuerpo'> y <cuerpo''> representan el conjunto de predicados de actualización situados a la izquierda y a la derecha respectivamente del predicado modificado. Ambos pueden ser vacíos.
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que el predicado ha sido modificado correctamente.
Ejecutada_por	Autor.

6.6 Conjunto de acciones evolutivas

A continuación, se resumen en la tabla 13, todas las acciones evolutivas aplicables sobre las reglas de actualización, contemplando para cada una: nombre, utilidad e implicaciones y quién puede ejecutarlas.

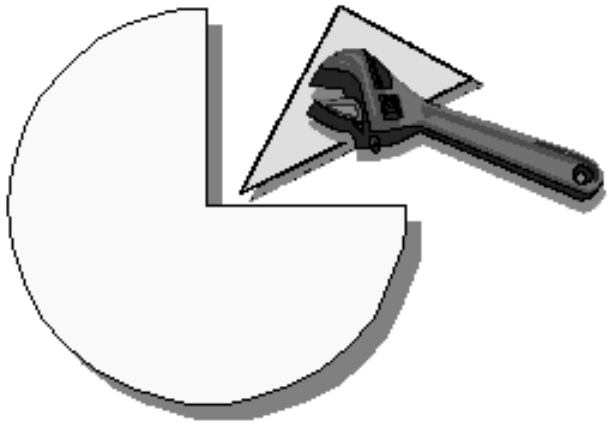
Tabla 13: Acciones evolutivas para modificar las reglas de actualización

ACe[Ru]	Utilidad/Implicaciones	
CrearRu	Se utiliza para crear la regla de actualización por defecto de un nuevo ítem i_j incluido en la presentación sobre la que se define EC_A^i. Esto implica que una nueva regla Ru(i_j) se añade al conjunto de reglas de actualización Ru(EC_A^i).	Sistema
EliminarRu	Se utiliza para eliminar la regla de actualización de un ítem i_j que es ocultado en la presentación asociada a la EC_A^i. Esto implica que la regla Ru(i_j) sea eliminada del conjunto de reglas de actualización Ru(EC_A^i).	

AñadirPredicadoRu	Se utiliza para especificar algún tipo de actualización sobre el ítem i_k como consecuencia de la visita del ítem i_j, eso sí, en posesión del conocimiento previo requerido. Esto implica que se añada un nuevo predicado de actualización asociado al ítem i_k al final del cuerpo de la regla $Ru(i_j)$.	Autor
ModificarPredicadoRu	Se utiliza para modificar los valores o el tipo de actualización definido para el ítem i_k en el cuerpo de la regla de actualización $Ru(i_j)$. Si i_k coincide con i_j la modificación sólo es realizada si se ajusta a lo permitido (sección 5.2.1). Esto implica que se sustituya en el cuerpo de $Ru(i_j)$ el predicado de actualización existente sobre i_k por un nuevo predicado definido también para dicho ítem.	
EliminarPredicadoRu	Se utiliza para suprimir la actualización de un ítem i_k tras la visita, según condiciones necesarias, del ítem i_j. Esto implica la eliminación del predicado de actualización definido para i_k en el cuerpo de la regla $Ru(i_j)$.	Ambos

CAPÍTULO 10

Consistencia Rk ∪ Ru



Resumen

Se hace necesario garantizar la alcanzabilidad de todos los ítems durante la navegación de una estructura conceptual de aprendizaje. Con este objetivo, se establece un procedimiento que, dado un conjunto de reglas de conocimiento, es capaz de detectar, si los hay, los ítems inalcanzables. Dicho procedimiento se apoya en dos propiedades previamente garantizadas: la monotonía de las restricciones de accesibilidad y la garantía de que visitando repetidamente un ítem accesible éste se conoce totalmente. Algunas situaciones de inalcanzabilidad provocadas por las reglas de conocimiento pueden resolverse con la adquisición de conocimiento propuesta en las reglas de actualización. Con este motivo, se justifica y define un segundo procedimiento que comprueba la alcanzabilidad analizando conjuntamente las reglas de conocimiento y actualización. Se estudian los casos en que dicha comprobación debe ser realizada, planteando la inclusión de poscondiciones en las acciones evolutivas especificadas en los dos capítulos anteriores. Para terminar se estudia la complejidad temporal y espacial que requiere la comprobación de alcanzabilidad.

Tabla de contenidos

Consistencia Rk ∪ Ru

1. INTRODUCCIÓN

En los capítulos anteriores se ha descrito formalmente la sintaxis y la semántica de las reglas de conocimiento (capítulo 8) y de las reglas de actualización (capítulo 9). Durante la definición de una regla, de cualquiera de estos dos tipos, el autor debe ajustarse a una serie de condiciones que tienen como objetivo garantizar la corrección no sólo de la regla individual, sino también, del conjunto de reglas donde ésta se integra.

Entre estas condiciones, existen algunas destinadas a poder garantizar la alcanzabilidad de todos los ítems incluidos en la estructura de navegación donde se define el conjunto de reglas de actualización y conocimiento (Rk ∪ Ru).

> **Def 10.1 [Ítem alcanzable]** Un ítem i_k incluido en una estructura de aprendizaje EC_A^i, es alcanzable si a través de la navegación de ésta, cualquier usuario puede conseguir un estado de conocimiento para el cuál se satisfacen las restricciones de accesibilidad de al menos una de las reglas de conocimiento, $Rk(i_k)$, asociadas a i_k en $Rk(EC_A^i)$.

Por lo tanto, dada una estructura conceptual de aprendizaje $EC_A^i = (EC_M, Rw, MU, EC_P^j, RTnb^k, Ro^k, Ru^i, Rk^i)$, el conjunto de reglas $Rk(EC_A^i) \cup Ru(EC_A^i)$ debe ser tal que se garantice la alcanzabilidad de todos los ítems mostrados en la presentación EC_P^j. Dicho de otro modo, permitir que un usuario que navega en una estructura de aprendizaje, EC_A^i, tenga en algún momento la capacidad de acceder al contenido de todos los ítems de información que en ésta se proporcionan.

En definitiva se trata de *evitar la existencia de ítems inalcanzables* para el usuario. En cada momento existe un conjunto de ítems accesibles que el usuario puede visitar de acuerdo a las restricciones de accesibilidad impuestas por el autor en las reglas de conocimiento. ¿Qué ítems pertenecen a ese conjunto?, depende directamente del estado de conocimiento que posee el usuario en ese preciso instante. Lo que se debe garantizar es que, a través de un adecuado proceso de navegación, el usuario tiene en su mano alcanzar un estado de conocimiento para el cual todos los ítems de la estructura navegada pertenecen al conjunto, es decir, son accesibles.

Para poder asegurar la propiedad de alcanzabilidad, en las reglas de conocimiento las condiciones impuestas por el autor para la visita de un ítem se separan internamente en dos tipos de restricciones: accesibilidad ($K(i_j) \geq$valor) e idoneidad ($K(i_j) \leq$valor). Únicamente las restricciones del primer tipo se evalúan para determinar la accesibilidad de un ítem, entre otros motivos (ver capítulo 8), porque las restricciones de idoneidad no son monótonas.

Esto es, si en un determinado instante el usuario satisface una restricción de idoneidad, por ejemplo $K(i_j) \leq 2$, no se asegura que ésta se siga satisfaciendo en todo instante posterior del proceso de navegación del usuario. De modo que si fuese utilizada para establecer la accesibilidad de un ítem, esté pasaría de ser accesible a inaccesible para el usuario cuando su conocimiento sobre i_j aumentase por encima de 2.

Al contrario, cuando una restricción de accesibilidad se satisface, por ejemplo $K(i_j) \geq 2$, se asegura que ésta se sigue satisfaciendo en todo instante posterior, ya que el conocimiento del usuario acerca de i_j no puede más que aumentar o mantenerse. De esta forma, cuando un ítem se vuelve accesible para el usuario sigue siéndolo de ahí en adelante. Esta propiedad de monotonía es necesaria para garantizar la existencia de un instante futuro a partir del cual todos los ítems son accesibles.

Por otro lado, el conjunto de reglas de actualización garantiza que si un usuario estudia un ítem en posesión del conocimiento previo exigido para asimilar correctamente su contenido, necesita un número finito de lecturas para alcanzar un conocimiento "total" sobre dicho ítem. Para ello, el sistema restringe los predicados de actualización que el autor puede aplicar al ítem de la cabeza en una regla de actualización, permitiendo sólo aquellos cuya ejecución repetida logra un grado de conocimiento "total" sobre el ítem actualizado (ver capítulo 9).

Partiendo de las dos condiciones anteriores es posible comprobar y garantizar una propiedad de navegación bastante deseable: *que cualquier usuario que navega una estructura de aprendizaje* EC_A^i*, pueda adquirir un grado de conocimiento "total" sobre todo ítem incluido en ésta*. Para que se satisfaga esto, el conjunto de reglas de actualización y conocimiento definidas en EC_A^i, debe satisfacer los dos requisitos explicados arriba:

a) la visita repetida a un ítem accesible debe proporcionar conocimiento "total" sobre éste, y

b) existe un instante a partir del cuál todos los ítems son accesibles para el usuario.

La satisfacción de la propiedad a) queda garantizada, por su propia construcción, en el conjunto de reglas de actualización, $Ru(EC_A^i)$. Para garantizar la satisfacción de la propiedad b) debemos comprobar y exigir la consistencia del conjunto de reglas de conocimiento y actualización, $Rk(EC_A^i) \cup Ru(EC_A^i)$. Como se explica en la siguiente sección puede exigirse el cumplimiento de b) sobre el conjunto de reglas de conocimiento, independientemente de las reglas de actualización. Sin embargo, este último modo es algo más restrictivo.

2. COMPROBAR ALCANZABILIDAD EN RK

Se pretende comprobar si dado un conjunto de reglas de conocimiento, $Rk(EC_A^i)$, todos los ítems de la estructura para la que se definen pueden ser visitados en algún momento durante la navegación del usuario. El autor durante el proceso de definición de dichas reglas puede haber creado, sin darse cuenta, situaciones contradictorias que hagan a un ítem o a un conjunto de ítems inaccesibles para el usuario, independientemente de la navegación que éste realice. Este tipo de ítems son inalcanzables y por lo tanto deben ser detectados y evitados durante la creación y modificación de las reglas de conocimiento.

Por ejemplo, imaginemos que la única manera de poder visitar el ítem I4 es teniendo sobre I3 un conocimiento mayor o igual que "medio", y que al mismo tiempo para poder acceder a I3 es necesario como mínimo un conocimiento "alto" sobre I4. Esto es,

el autor ha definido para los ítems I3 e I4 las reglas de conocimiento que se muestran en la tabla 1.

Tabla 1: Reglas de conocimiento para I3 e I4

Rk(I3) : [K(I4) ≥ 3] *and‴* true → Visitar(I3)
Rk(I4) : [K(I3) ≥ 2] *and‴* true → Visitar(I4)

Asumiendo las reglas de actualización por defecto, si nuestro usuario no satisface antes de navegar en esta estructura, al menos una de las condiciones de conocimiento impuestas, ya no tendrá manera de satisfacerlas durante su navegación en la misma. Esto ocurre porque existe una situación de **interbloqueo** entre I3 e I4, que convierte ambos ítems en inalcanzables.

Para poder comprobar si el conjunto de reglas de conocimiento definidas por el autor en una estructura de aprendizaje, $EC_A^i = (EC_M, Rw, MU, EC_P^j, RTnb^k, Ro^k, Ru^i, Rk^i)$, garantiza la alcanzabilidad del conjunto completo de los ítems mostrados en EC_P^j, se propone un algoritmo que permite detectar la existencia de ítems inalcanzables.

Las premisas en las que se apoya el algoritmo son las siguientes:

a) El conjunto de ítems accesibles es vacío al iniciar la ejecución del algoritmo.

$$A = \varnothing$$

b) Si existe una regla de conocimiento para i_1 donde el antecedente de accesibilidad es *true*, entonces i_1 es accesible.

$$\text{true} \rightarrow \text{Accesible}(i_1) \;\Rightarrow\; A = A \cup \{i_1\}$$

c) Si un ítem i_1 es accesible se puede alcanzar conocimiento "total" sobre él. Por lo tanto, se satisface cualquier restricción de accesibilidad impuesta para éste en una regla de conocimiento.

$$i_1 \in A \;\Rightarrow\; \forall \text{Restricción}^A(i_1) \text{ es true}$$

d) Si existe una regla de conocimiento para i_2 donde en el antecedente de accesibilidad se exige únicamente una restricción sobre i_1, y i_1 es accesible, se deriva de c) que i_2 es también accesible.

$$\text{Restricción}^A(i_1) \rightarrow \text{Accesible}(i_2) \text{ y } i_1 \in A \;\Rightarrow\; A = A \cup \{i_2\}$$

e) Si existe una regla de conocimiento para i_n donde en el antecedente de accesibilidad se exigen dos o más restricciones, todas ellas sobre ítems accesibles, entonces se deriva de d) que i_n es también accesible.

$$\text{Restricción}^A(i_1) \wedge \text{Restricción}^A(i_2) \wedge \ldots \rightarrow \text{Accesible}(i_n)$$
$$\text{y } i_1, i_2, \ldots \in A \;\Rightarrow\; A = A \cup \{i_n\}$$

La alcanzabilidad debe garantizarse para un usuario con un estado de desconocimiento absoluto, por eso la aplicación del algoritmo supone un grado de conocimiento inicialmente "nulo" para cada ítem.

Paso 0. Hacer una copia, $Rk'(EC_A^i)$, del conjunto de reglas de conocimiento $Rk(EC_A^i)$.

Paso 1. Inicializar el conjunto de ítems accesibles (A).

$A = \{ i_j / i_j$ es inicialmente accesible$\}$

Un ítem i_j es inicialmente accesible si:

a) No tiene ninguna regla de conocimiento asociada, esto es: $Rk'(i_j) = \varnothing$ o

b) En el conjunto de sus reglas de conocimiento $Rk'(i_j) = \{Rk'^1(i_j), Rk'^2(i_j), ..., Rk'^n(i_j)\}$ existe al menos una regla $Rk'^s(i_j)$, con s = 1. .n, donde el antecedente de accesibilidad es vacío, esto es: $Rk'^s(i_j)$: true *and''* Antecedente_Idoneidad → Visitar(i_j).

Paso 2. Evaluar en las reglas de conocimiento $Rk'(i_k)$ asociada a un ítem i_k no accesible, las restricciones de accesibilidad que están satisfechas, considerando para ello conocimiento "total" sobre cada ítem i_j accesible. Esto es:

Para todo i_k tal que $i_k \notin A$: Eliminar en el antecedente de accesibilidad de cada una de sus reglas de conocimiento $Rk'^1(i_k)$, $Rk'^2(i_k)$, ..., $Rk'^n(i_k)$ las restricciones de conocimiento impuestas sobre un ítem $i_j \in A$.

Si se produce alguna modificación en el conjunto de reglas de conocimiento Rk': Ir al paso 3, en otro caso: Ir al paso 4.

Paso 3. Detectar nuevos ítems accesibles después de los cambios producidos en las reglas de conocimiento.

$A = A \cup \{i_k / i_k$ se ha convertido en accesible$\}$

Un ítem i_k se convierte en accesible si el antecedente de accesibilidad de alguna de sus reglas de conocimiento $Rk'^1(i_k)$, $Rk'^2(i_k)$, ..., $Rk'^n(i_k)$ se ha quedado vacío (es *true*) después de eliminar una o varias restricciones de conocimiento en el paso 2.

Si se produce alguna modificación en el conjunto de ítems accesibles, A: Ir al paso 2, en otro caso: Ir al paso 4.

Paso 4. Comprobar si todos los ítems son accesibles.

Siendo $I(EC_P^j)$ el conjunto de todos los ítems mostrados en la estructura conceptual de presentación EC_P^j para la que se define el conjunto de reglas de conocimiento $Rk(EC_A^i)$.

Si $I(EC_P^j) - A = \varnothing$ se garantiza que todos los ítems de la estructura son accesibles en algún momento, por lo tanto el conjunto de reglas de conocimiento es válido.

En otro caso, si $I(EC_P^j) - A = \{i_1, i_2,..., i_m\}$ el sistema ha detectado que dado el conjunto de reglas de conocimiento $Rk(EC_A^i)$ existe un conjunto de ítems inalcanzables: $\{i_1, i_2,... , i_m\}$.

Terminar.

Algoritmo 1. Comprobar alcanzabilidad en $Rk(EC_A^i)$

Adviértase que la verificación de accesibilidad que realiza el algoritmo se basa principalmente en la satisfacción de dos propiedades previamente garantizadas:

i) Monotonía de la accesibilidad de un ítem: A partir del momento en que un ítem se vuelve accesible para un usuario se mantiene así a medida que éste navega y aumenta su estado de conocimiento.

ii) Conocimiento "total" de un ítem tras la ejecución repetida de su regla de actualización: La visita recurrente al contenido de un ítem garantiza la consecución de un conocimiento "total" sobre éste.

La propiedad i) es la responsable de que cuando un ítem i_j entra a formar parte del conjunto de accesibilidad A (pasos 1 y 3 del algoritmo) permanezca en él durante las siguientes iteraciones de los pasos 2 y 3 del algoritmo.

La propiedad ii) es la que permite asegurar que cuando un ítem i_j es accesible, $i_j \in A$, cualquier restricción de accesibilidad sobre éste, $K(i_j) \geq$valor, impuesta en el cuerpo de la regla de conocimiento de otro ítem i_k puede ser satisfecha. Ya que, si es posible obtener conocimiento "total" sobre i_j está garantizado el cumplimiento de la restricción de accesibilidad $K(i_j) \geq$*valor* para cualquier *valor*.

Se demuestra que el algoritmo propuesto es correcto y completo. No puede ser introducido en *A* un ítem que no sea accesible, ya que esto significaría que cuando acaba el algoritmo en todas sus reglas de conocimiento existe al menos una restricción de accesibilidad no satisfecha. Del mismo modo, un ítem accesible no puede dejar de ser introducido en *A* porque en alguna iteración del algoritmo todas las restricciones de accesibilidad de alguna de sus reglas han sido satisfechas.

3. COMPROBAR ALCANZABILIDAD EN RK ∪ RU

El algoritmo descrito en la sección anterior considera el conjunto de reglas de actualización creadas inicialmente por el sistema. Estas reglas actualizan únicamente el ítem visitado, sin modificar para nada el grado de conocimiento que el usuario posee sobre el resto de los ítems. Sin embargo, tal y como se explicó en el capítulo 9 (reglas de actualización), el autor puede modificar la regla de actualización por defecto de un ítem, incluyendo en su cuerpo la actualización de algunos ítems cuyo contenido está muy relacionado con el del ítem visitado.

De esta forma, pueden existir en las reglas de conocimiento, algunas restricciones de accesibilidad asociadas a ítems aún no accesibles, que sin embargo, pueden ser satisfechas mediante la visita o visitas de uno o varios ítems relacionados. Es decir, la restricción de accesibilidad $K(i_j) \geq$valor no se satisface mediante la ejecución repetida de la regla de actualización de i_j (ya que actualmente i_j no es accesible), sino mediante la ejecución de otras reglas asociadas a ítems sí accesibles, en cuyo cuerpo se incluye un predicado de actualización definido sobre i_j.

Retomando las reglas de conocimiento de I3 e I4 mostradas en el ejemplo anterior (tabla 1), y suponiendo nuevamente que el usuario no satisface, de forma previa, las restricciones de accesibilidad de ninguna de ellas. Es fácil ver como la situación de interbloqueo se resuelve si, por ejemplo, la regla de actualización asociada a un ítem I2, accesible actualmente para el usuario, es la que se muestra en la tabla 2.

Tabla 2: Regla de actualización para I2

Ru(I2) : Si Visitado(I2) entonces Fix-abs(I2, "total"), Fix-abs(I3, "alto");

Si nuestro usuario visita I2 obtiene conocimiento "alto" sobre I3 sin necesidad de visitarlo. Esto hace que la restricción de accesibilidad, K(I3) ≥ 2("medio") necesaria para visitar I4 sea satisfecha, y por lo tanto, que I4 se vuelva accesible. Después, visitando una o más veces I4, el conocimiento del usuario acerca de éste se hace "total", satisfaciéndose como consecuencia la restricción necesaria para que I3 sea accesible.

3.1 Descripción de la problemática

Para comprobar la alcanzabilidad de todos los ítems teniendo en cuenta, además de las reglas de conocimiento, el conjunto de reglas de actualización es necesario definir un nuevo algoritmo. A continuación se exponen algunos de los aspectos más complicados relacionados con el diseño de este algoritmo.

Cuando un ítem se vuelve accesible está garantizado que es posible alcanzar conocimiento "total" sobre dicho ítem. La dificultad radica en determinar *cuál es el grado de conocimiento máximo que la visita repetida a ese ítem accesible, i_j, puede lograr sobre cada uno de los ítems i_k actualizados en el cuerpo de su regla de actualización*. Dicho conocimiento máximo depende del tipo de actualización definida sobre i_k, pero si ésta es relativa, en algunos casos depende también de la actualización realizada sobre el propio i_j. La siguiente tabla se describe conceptualmente cómo se determina este conocimiento máximo.

Tabla 3: Conocimiento máximo para i_k visitando i_j

Ru(i_j): Si Visitado(i_j) entonces Actualización(i_j), ..., Actualización(i_k), ...;	
Actualización(i_k)	**Conocimiento máximo sobre i_k**
Fix-abs(i_k, val)	Una actualización "fija" y "absoluta" sólo produce modificación en el conocimiento del ítem para el que se define la primera vez que ésta se lleva a cabo. Por tanto, el conocimiento máximo que se puede alcanzar sobre el ítem i_k visitando i_j es el valor al que se fija en el predicado Fix-abs, esto es, *val*.
Fix-abs$_{first}$(i_k, val)	
Fix-rel(i_k)	Una actualización "fija" y "relativa" de tipo "cada-vez" permite obtener sobre i_k el mismo grado de conocimiento que se obtiene con la visita repetida a i_j. Puesto que visitando varias veces i_j se alcanza conocimiento "total", el conocimiento máximo que se obtiene sobre i_k visitando i_j es también "total".
Fix-rel$_{first}$ (i_k)	En una actualización "fija" y "relativa" de tipo "primera-vez", el conocimiento máximo sobre i_k coincide con el conocimiento máximo que es posible obtener sobre i_j la primera vez que éste se visita. Es necesario considerar las dos posibles actualizaciones para la cabeza: - Si Actualización(i_j) es Fix-abs(i_j, "total"), el conocimiento sobre i_k después de visitar una vez i_j es también "total". - Si Actualización(i_j) es Inc-abs(i_j, *val*) el conocimiento máximo sobre i_k es igual al conocimiento máximo que es posible obtener sobre i_j aún no visitado más el valor de incremento, *val*, especificado en la actualización de i_j tras su visita.

Inc-abs(i_k, val)	Una actualización "incremental" y "absoluta" de tipo "cada-vez" antes o después termina alcanzando conocimiento máximo sobre el ítem actualizado. Por lo tanto, visitando repetidamente i_j se puede alcanzar conocimiento "total" sobre i_k.
Inc-abs$_{first}$ (i_k, val)	Una actualización "incremental" y "absoluta" de tipo "primera vez" sólo es ejecutada tras la primera visita a i_j. Permitiendo alcanzar sobre el ítem i_k un grado de conocimiento igual al conocimiento máximo que es posible obtener sobre él visitando otros ítems accesibles distintos de i_j más el incremento (*val*) especificado.
Inc-rel(i_k, val)	Una actualización "incremental" y "relativa" de tipo "cada vez" permite alcanzar sobre i_k un conocimiento igual al grado de conocimiento máximo sobre i_j más/menos el incremento/decremento especificado en *val*. Puesto que visitando repetidamente i_j se garantiza conseguir conocimiento "total" sobre éste, el conocimiento máximo sobre i_k será: - El grado de conocimiento "total" si *val* es positivo. - El grado de conocimiento "total" reducido en los niveles indicados en *val*, cuando *val* es negativo.
Inc-rel$_{first}$ (i_k, val1)	Cuando la actualización es "incremental" y "relativa" de tipo "primera-vez" es necesario hacer distinción entre los dos posibles predicados de actualización para i_j: - Si Actualización(i_j) es Fix-abs(i_j, "total"), el conocimiento sobre i_k después de visitar por primera vez i_j es: ▪ El grado de conocimiento "total" si $0<val1\leq 4$, ó ▪ El grado de conocimiento "total" reducido en el numero de grados indicados en *val1* si $-4 < val1 < 0$. - Si Actualización(i_j) es Inc-abs(i_j, *val2*), el conocimiento máximo sobre i_k es igual al grado de conocimiento máximo sobre i_j antes de ser visitado más el incremento *val2* indicado para i_j, y: ▪ Más el incremento *val1* especificado en la actualización de i_k si $0<val1\leq 4$, ó ▪ Menos el decremento *val1* especificado para i_k si $-4 < val1 < 0$.

Como puede observarse en la tabla 3, el grado de conocimiento máximo que puede alcanzarse sobre un ítem i_k visitando un ítem accesible i_j es "total" si la actualización es de tipo "cada-vez", salvo claro está, que se trate de una actualización "fija" y "absoluta" a un valor distinto de "total" o de un decremento relativo. En cualquier caso, si la actualización es de tipo "cada-vez" obtener el conocimiento máximo sobre i_k es inmediato. En el caso de actualizaciones "relativas", esto es posible gracias a la propiedad de las reglas de actualización que garantiza la consecución de conocimiento "total" sobre un ítem i_j ejecutando de forma repetida su regla de actualización Ru(i_j).

Sin embargo, cuando la actualización sobre i_k es de tipo "primera-vez", exceptuando el caso del predicado Fix-abs$_{first}$, el cálculo del conocimiento máximo sobre i_k visitando una sola vez i_j no es tan trivial:

a) Si la actualización definida sobre i_k es un "incremento" "absoluto" es necesario conocer cuál es el grado de conocimiento máximo que es posible alcanzar sobre i_k sin haber visitado previamente i_j y luego sumarle el número de grados indicado.

b) Si el predicado de actualización para i_k es "relativo" es necesario averiguar el conocimiento máximo sobre i_j visitándolo sólo una vez. Esto plantea dos situaciones diferentes:

   b.1) cuando la actualización sobre i_j es "fija", esto es Fix-abs(i_j, "total") el problema se resuelve fácilmente,

   b.2) pero cuando se trata de un "incremento", esto es Inc-abs(i_j, val), es necesario averiguar el máximo conocimiento que es posible obtener sobre i_j sin necesidad de visitarlo para luego aplicarle el incremento indicado.

La forma de obtener la información necesaria en a) y b.2) es evaluar las reglas de actualización que en su cuerpo definen una actualización para i_k o i_j respectivamente. Ahora bien, teniendo en cuenta, sólo aquellas reglas asociadas a un ítem actualmente accesible, ya que el resto no pueden ser ejecutadas hasta que el ítem de la cabeza se vuelva accesible.

Llegado a este punto se nos plantea otra cuestión complicada. El conjunto de ítems accesibles varía a medida que se visitan nuevos ítems y se ejecutan sus reglas de actualización, dando paso a un nuevo estado de conocimiento donde se satisfacen las restricciones de accesibilidad de algunos ítems para los que antes esto no ocurría. Por lo tanto, el conocimiento máximo que es posible alcanzar sobre i_j o i_k antes de visitar i_j depende del conjunto de ítems accesibles en ese momento.

Cuando se amplía el conjunto de ítems accesibles la cuestión es: ¿Debemos reevaluar el conocimiento máximo sobre los ítems i_k que presenten una actualización de tipo "primera-vez" en el cuerpo de la regla Ru(i_j)? La respuesta es negativa si queremos asegurar que, sea cual sea la navegación que realiza el usuario, puede llegar a alcanzar todos los ítems con conocimiento "total".

Para ilustrar la conclusión anterior, a la que llegamos analizando en detalle todas las situaciones posibles, recurriremos a un ejemplo que nos ayuda a clarificar una situación concreta, antes de exponer la problemática de forma general.

Supongamos el conjunto de reglas de conocimiento expresado en la tabla 4 y las reglas de actualización de la tabla 5. En el conjunto de reglas de conocimiento existe una situación de interbloqueo entre los ítems I4 e I5. Se trata de evaluar si esta situación puede ser resuelta mediante las reglas de actualización.

Tabla 4: Reglas de conocimiento

Rk(I1) : [K(I2) ≥ 2] and''' true → Visitar(I1)
Rk(I3) : [K(I2) ≥ 1] and''' true → Visitar(I3)
Rk(I4) : [K(I5) ≥ 4] and''' true → Visitar(I4)
Rk(I5) : [K(I4) ≥ 3] and''' true → Visitar(I5)

Tabla 5: Reglas de actualización

Ru(I1) : Si Visitado(I1) entonces Inc-abs(I1, 1), Fix-rel$_{first}$(I5);
Ru(I2) : Si Visitado(I2) entonces Inc-abs(I2, 2), Fix-abs(I1,"medio");
Ru(I3) : Si Visitado(I3) entonces Fix-abs(I3, "total"), Inc-abs(I1, 1);
Ru(I4) : Si Visitado(I4) entonces Fix-abs(I4, "total");
Ru(I5) : Si Visitado(I5) entonces Fix-abs(I5, "total");

Vamos a demostrar que si se reevalúan los predicados de tipo "primera-vez", lo único que se puede garantizar es que existe al menos un camino de navegación donde todos los ítems son alcanzables. De modo que para asegurar que cualquier camino de navegación cumple esta propiedad los predicados de tipo "primera-vez" se deben evaluar sólo una vez, concretamente en cuanto el ítem cabeza de la regla se hace accesible.

Paso 1. Suponemos un estado inicial de desconocimiento absoluto.

En esta situación el único ítem accesible es I2. De acuerdo a la tabla 3, visitando repetidamente I2 se obtiene un conocimiento máximo "total" para I2 y "medio" para I1.

Paso 2. Considerando ese conocimiento máximo, los ítems I1 e I3 se vuelven accesibles.

Paso 2.1 Evaluando Ru(I1) según la tabla 3, el conocimiento máximo sobre I1 es "total". Y el conocimiento máximo sobre I5 tras visitar una vez I1 es "alto", lo cual corresponde al conocimiento máximo sobre I1 visitando únicamente I2 ("medio") más un grado de incremento.

Paso 2.2 Evaluando Ru(I3) según la tabla 3, el conocimiento máximo sobre I3 e I1 es "total".

Paso 3. Las modificaciones de conocimiento del paso 2 no hacen accesible I4 ni I5 que por lo tanto, se identifican como ítems inalcanzables.

Sin embargo, si al detectar en el paso 2.2 un aumento en el conocimiento máximo posible para I1 sin visitarlo, se reconsidera el predicado Fix-rel$_{first}$(I5) existente en la regla Ru(I1), el conocimiento máximo sobre I5 visitando una vez I1 sería "total" con lo que I4 se convierte en accesible y se rompe el interbloqueo.

El problema, es que esta reevaluación obliga a que I1 se visite necesariamente después que I3, cuando en realidad los dos ítems son accesibles en el mismo momento y el usuario podría visitarlos en cualquiera de los dos ordenes posibles: I1, I3 o I3, I1.

En general, cuando en un mismo paso se hacen accesibles un conjunto de ítems $\{I_1,..., I_n\}$ no se debe imponer ningún orden para visitarlos. Para conseguirlo, durante la evaluación de la regla de actualización $Ru(I_j)$ no se tiene en cuenta el resultado de evaluar ninguna de las reglas $Ru(I_1)$, ..., $Ru(I_{j-1})$ evaluadas anteriormente, ya que eso equivale a exigir que el ítem I_j sea visitado después que los ítems I_{j-i}. Tampoco al evaluar una regla posterior $Ru(I_{j+1})$, ..., $Ru(I_n)$ se volverá a reconsiderar $Ru(I_j)$, ya que esto supone exigir que la visita de I_j se realice después de visitar I_{j+i}.

Por supuesto, tampoco en el paso siguiente, como consecuencia de la evaluación de la regla $Ru(I_t')$ asociada a un nuevo ítem accesible I_t' se volverá a evaluar la regla $Ru(I_j)$, ya que esto significaría obligar a que el ítem I_t' sea visitado antes que I_j cuando este último se convierte en accesible antes, e incluso puede ser responsable en parte de la reciente accesibilidad de I_t'.

3.2 Descripción del algoritmo

A continuación proponemos un algoritmo que, teniendo en cuenta todo lo explicado en la sección 3.1, permite detectar la existencia de ítems inalcanzables, y en ausencia de éstos, garantizar la alcanzabilidad de todos los ítems sea cual sea el orden en que el usuario visita los ítems accesibles. Esta propiedad es muy importante ya que asegura que independientemente de las visitas realizadas hasta el momento, el usuario puede alcanzar cualquier ítem (antes o después según la eficacia de su proceso de navegación), adecuando sus visitas al único orden que le imponen las propias reglas de conocimiento.

Las premisas en las que se apoya el algoritmo son las siguientes:

a) El conjunto de ítems accesibles es vacío al iniciar el algoritmo.

$$A = \varnothing$$

b) Si un ítem i_1 no es accesible, el máximo conocimiento que se puede obtener con su visita sobre cualquier ítem i_j, notado como $MáximoK[i_1, i_j]$, es 0.

$$i_1 \notin A \Rightarrow \forall_j \; MáximoK[i_1, i_j] = 0$$

c) Si existe una regla de conocimiento para i_1 donde el antecedente de accesibilidad es *true*, entonces i_1 es accesible.

$$true \rightarrow Accesible(i_1) \Rightarrow A = A \cup \{i_1\}$$

d) Al hacerse un ítem i_1 accesible permite alcanzar un determinado conocimiento máximo sobre él y el resto de ítems actualizados en su regla de actualización.

$Ru(i_1)$: Si Visitado(i_1) entonces Actualización(i_1), Actualización(i_2), ...
..., Actualización(i_n);

Cuando i_1 se hace accesible $\Rightarrow$ Se calcula $MáximoK[i_1, i_1]$, $MáximoK[i_1, i_2]$, ...
..., $MáximoK[i_1, i_n]$ como se justifica en la tabla 3.

e) El máximo conocimiento que se puede alcanzar sobre un ítem i_2, notado como $MáximoK[i_2]$, es el conocimiento más alto que actualmente algún ítem i_j permite alcanzar sobre él.

$$MáximoK[i_2] = máximo_j \; (MáximoK[i_j, i_2])$$

f) Una restricción de accesibilidad $Restricción^A(i_2)$ impuesta sobre un ítem i_2 se satisface si el conocimiento máximo que se puede obtener sobre i_2 supera o iguala el valor exigido.

$$\text{MáximoK}[i_2] \geq \text{valor} \Rightarrow \text{Restricción}^{A}(i_2) \equiv K(i_2) \geq \text{valor es } \textit{true}$$

Finalmente el algoritmo propuesto es el siguiente:

Paso 0. Hacer una copia $Rk'(EC_A^i)$ del conjunto de reglas de conocimiento $Rk(EC_A^i)$.

Paso 1. **Inicializaciones.**

Paso 1.1 Inicializar el conjunto de ítems accesibles como vacío, esto es A = ∅.

Paso 1.2 Inicializar el conjunto A_N que contendrá en cada iteración los ítems convertidos en accesibles como resultado de la anterior iteración del algoritmo. En la primera iteración un ítem i_i forma parte del conjunto A_N si es inicialmente accesible (suponiendo un estado de desconocimiento absoluto).

Un ítem i_j es inicialmente accesible, y por lo tanto $i_j \in A_N$, sólo si:

a) No tiene ninguna regla de conocimiento asociada ($Rk'(i_j) = \emptyset$) ó

b) En el conjunto de sus reglas de conocimiento $Rk'(i_j) = \{Rk'^1(i_j), Rk'^2(i_j), ..., Rk'^n(i_j)\}$ existe al menos una, $Rk'^s(i_j)$, donde el antecedente de accesibilidad es vacío, esto es: $Rk'^s(i_j)$: true *and‴* Antecedente_Idoneidad → Visitar(i_j).

Paso 1.3 Crear la matriz M_k donde se almacenará el grado máximo de conocimiento que la visita a un ítem accesible i_i puede lograr sobre los ítems i_k actualizados en su regla de actualización.

Sea *n* el número de ítems mostrados en la estructura conceptual de presentación EC_P^j a la que se asocia el conjunto de reglas verificado, esto es $EC_A^i = (EC_M, Rw, MU, EC_P^j, RTnb^k, Ro^k, Ru^i, Rk^i)$. La matriz M_k tiene dimensión *n*×*n.* La celda $M_k[j,k]$ contiene en cada momento el grado de conocimiento máximo que la ejecución (repetida si es posible) de la regla $Ru(i_j)$ puede lograr sobre el ítem i_k actualizado en su cuerpo.

Paso 1.4 Inicializar todas las celdas de la matriz M_k a cero. $\forall j = 1..n, k = 1..n : M_k[j,k] = 0$.

Paso 2. **Actualizar la matriz M_k.**

Paso 2.1 Hacer una copia de M_k en M_k', esto es $\forall_{i,j}\ M_k'[i,j] = M_k[i,j]$.

Paso 2.2 Actualizar las celdas de M_k' en función de los nuevos ítems accesibles.

Para cada ítem i_i incluido en el conjunto de nuevos ítems accesibles, esto es $\forall i_i \in A_N$: Tomar su regla de actualización $Ru(i_i)$ y para cada ítem i_k actualizado en su cuerpo actualizar la celda $M_k'[j,k]$ tal y como se indica en la tabla 6.

Adviértase que, en aquellas ocasiones que se requiere, para el cálculo de $M_k'[j,k]$ se utilizan los valores no actualizados de M_k, es decir las celdas $M_k[s,u]$ en lugar de $M_k'[s,u]$. De esta forma las modificaciones realizadas durante esta iteración no son tenidas en cuenta durante la misma.

Tabla 6: Conocimiento máximo para i_k visitando i_j

$Ru(i_j)$: Si Visitado(i_j) entonces Actualización(i_j), ..., Actualización(i_k), ...;		
Actualización(i_k)	**Actualización(i_j)**	**$M_k'[j, k]$**
Fix-abs(i_k, val)	_	$M_k'[j,k]$ = *val*.
Fix-abs$_{first}$(i_k, val)	_	$M_k'[j,k]$ = *val*.
Fix-rel(i_k)	_	$M_k'[j,k]$ = valor-numérico(“total”) = 4.

Fix-rel$_{first}$ (i_k)	Fix-abs(i_j, "total")	M_k'[j,k] = valor-numérico("total") = 4.
	Inc-abs(i_j, val)	M_k'[j,k] = $máximo_s(M_k[s, j]) + val$, donde el ítem i_s es distinto de i_j ($s \neq j$). Si el valor obtenido es mayor que 4 (M_k'[j,k] > 4), se normaliza a 4 (M_k'[j,k] = 4).
Inc-abs(i_k, val)	_	M_k'[j,k] = valor-numérico("total") = 4.
Inc-abs$_{first}$ (i_k, val)	_	M_k'[j,k] = $máximo_s(M_k[s,k]) + val$, donde el ítem i_s es distinto de i_j ($s \neq j$). Si el valor obtenido es mayor que 4 (M_k'[j,k] > 4), se normaliza a 4 (M_k'[j,k] = 4).
Inc-rel(i_k, val)	_	Si 0 < *val* ≤ 4 entonces: M_k'[j,k] = valor-numérico("total") = 4. Si -4 < *val* < 0 entonces: M_k'[j,k] = valor-numérico("total") - |*val*| = 4 - |*val*|.
Inc-rel$_{first}$ (i_k, val1)	Fix-abs(i_j, "total")	Si 0 < *val1* ≤ 4 entonces: M_k'[j,k] = valor-numérico("total") = 4. Si -4 < *val1* < 0 entonces: M_k'[j,k] = valor-numérico("total") - |*val1*| = 4 - |*val1*|. Cuando M_k'[j,k] es negativo (M_k'[j,k] < 0) se normaliza a 0 (M_k'[j,k] = 0).
	Inc-abs(i_j, val2)	Si 0 < *val1* ≤ 4 entonces: M_k'[j,k] = $máximo_s(M_k[s, j]) + val1 + val2$, donde el ítem i_s es distinto de i_j ($s \neq j$). Cuando el valor obtenido es mayor que 4 (M_k'[j,k] > 4), se normaliza a 4 (M_k'[j,k] = 4). Si -4 < *val1* < 0 entonces: M_k'[j,k] = $[máximo_s(M_k[s, j]) + val1] - \|val2\|$, donde el ítem i_s es distinto de i_j ($s \neq j$). Cuando el valor obtenido es mayor que 4 (M_k'[j,k] > 4), se normaliza a 4 (M_k'[j,k] = 4). Cuando M_k'[j, k] es negativo (M_k'[j,k] < 0) se normaliza a 0 (M_k'[j,k] = 0).

Paso 2.3 La matriz M_k' pasa a ser M_k. Esto es, $\forall_{i,j}$ M_k [i,j] = M_k'[i,j].

Paso 3. **Evaluar** en el cuerpo de cada **regla de conocimiento** $Rk'^{i}(i_t)$ asociada a un ítem i_t aún no accesible las restricciones de accesibilidad que se cumplen, considerando para ello los grados de conocimiento especificados en la matriz M_k actualizada en el paso anterior.

Para todo ítem i_t tal que $i_t \notin A$ y $i_t \notin A_N$: Eliminar en el antecedente de accesibilidad de cada una de sus reglas de conocimiento $Rk'^{1}(i_t)$, $Rk'^{2}(i_t)$,..., $Rk'^{n}(i_t)$ la restricción de accesibilidad $K(i_u) \geq val$ si se satisface que el valor exigido es menor o igual que el grado de conocimiento máximo actualmente alcanzable sobre i_u, esto es *val* ≤ $máximo_s(M_k[s,u])$.

Si se produce alguna modificación en el conjunto de reglas de conocimiento $Rk'(EC_A^i)$: Ir al paso 4, en otro caso: Ir al paso 5.

Paso 4. **Detectar nuevos ítems accesibles** después de los cambios producidos en las reglas de conocimiento $Rk'(EC_A^i)$.

Paso 4.1 Pasar los ítems del conjunto A_N al conjunto A.

$A = A \cup A_N$.

$A_N = \varnothing$.

Paso 4.2 Obtener el conjunto de nuevos ítems accesibles A_N en esta iteración.

$A_N = \{i_k / i_k$ se ha convertido en accesible$\}$

Un ítem i_k se convierte tras el paso 3 en accesible si el antecedente de accesibilidad de alguna de sus reglas de conocimiento $Rk'^1(i_k)$, $Rk'^2(i_k)$,..., $Rk'^n(i_k)$ se ha quedado vacío (es *true*) después de eliminar una o varias restricciones de conocimiento. En ese caso $A_N = A_N \cup \{i_k\}$.

Si el conjunto de nuevos ítems accesibles es distinto de vacío, esto es $A_N \neq \varnothing$: Ir al paso 2 para actualizar la matriz M_k, en otro caso: Ir al paso 5.

Paso 5. **Comprobar si todos los ítems son accesibles**.

Sea $I(EC_P^j)$ el conjunto de todos los ítems mostrados en la estructura conceptual de presentación EC_P^j sobre la que se define la EC_A^i.

Si $I(EC_P^j) - A = \varnothing$, se garantiza que todos los ítems de la estructura conceptual de presentación son accesibles en algún momento, por lo tanto el conjunto de reglas de conocimiento y actualización es válido.

En otro caso, si $I(EC_P^j) - A = \{i_1, i_2,..., i_m\}$ el sistema ha detectado que dado el conjunto de reglas de conocimiento $Rk(EC_A^i)$ y de actualización $Ru(EC_A^i)$ existe un conjunto de ítems que no pueden ser siempre alcanzados: $\{i_1, i_2, ..., i_m\}$.

Terminar.

Algoritmo 2. Comprobar alcanzabilidad en $Rk(EC_A^i) \cup Ru(EC_A^i)$

La aplicación del algoritmo para el conjunto de reglas de conocimiento y actualización presentado en las tablas 4 y 5 es el siguiente:

Inicialización:

Rk'(I1) : [K(I2) ≥ 2] *and‴* true → Visitar(I1)
Rk'(I3) : [K(I2) ≥ 1] *and‴* true → Visitar(I3)
Rk'(I4) : [K(I5) ≥ 4] *and‴* true → Visitar(I4)
Rk'(I5) : [K(I4) ≥ 3] *and‴* true → Visitar(I5)

$A = \varnothing$.

I2 es el único ítem accesible inicialmente, por lo tanto $A_N = \{I2\}$.

M_k tiene dimensión 5×5 porque existen cinco ítems: I = {I1, I2, I3, I4 e I5}.

$$M_{k\,5x5} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Iteración 1:

Ru(I2) : Si Visitado(I2) entonces Inc-abs(I2, 2), Fix-abs(I1,"medio");

El máximo conocimiento que se puede conseguir con la visita de I2 es "medio" para I1 y "total" para I2. Esto es: $M_k'[2,1] = 2$ y $M_k'[2,2] = 4$.

$$M_{k\,5x5} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Rk'(I1) : true and''' true $\rightarrow$ Visitar(I1)
Rk'(I3) : true and''' true $\rightarrow$ Visitar(I3)
Rk'(I4) : [K(I5) ≥ "total"] and''' true $\rightarrow$ Visitar(I4)
Rk'(I5) : [K(I4) ≥ "alto"] and''' true $\rightarrow$ Visitar(I5)

A = {I2}.

En esta iteración se han hecho accesibles I1 y I3. A_N = {I1, I3}.

Iteración 2:

Ru(I1) : Si Visitado(I1) entonces Inc-abs(I1, 1), Fix-rel_{first}(I5);
Ru(I3) : Si Visitado(I3) entonces Fix-abs(I3, "total"), Inc-abs(I1, 1);

Se obtiene el conocimiento máximo tras la visita de I1 e I3. $M_k'[1,1] = 4$. $M_k'[1,5] = 3$. $M_k'[3,1] = 4$. $M_k'[3,3] = 4$.

$$M_{k\,5x5} = \begin{pmatrix} 4 & 0 & 0 & 0 & 3 \\ 2 & 4 & 0 & 0 & 0 \\ 4 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A pesar de las modificaciones en M_k no hay cambios en Rk'. $A_N = \varnothing$.

A = {I2, I1, I3}. I – A = {I4, I5}. I4 e I5 son detectados como inalcanzables.

4. ¿CUÁNDO COMPROBAR LA CONSISTENCIA DE RK ∪ RU?

El anterior algoritmo permite comprobar si el conjunto de reglas de conocimiento y actualización definido por el autor en una EC_A^i es consistente de acuerdo a la siguiente definición:

> **Def 10.2 [Consistencia Rk ∪ Ru]** Diremos que un conjunto de reglas $Rk(EC_A^i) \cup Ru(EC_A^i)$ es consistente si garantiza la alcanzabilidad de todos los ítems incluidos en la EC_A^i, con independencia de la navegación realizada hasta el momento y el estado de conocimiento inicial del usuario.

De esta forma se puede asegurar que un usuario no será privado de la posibilidad de alcanzar un ítem mostrado en la estructura que navega. Siendo por consiguiente, capaz de alcanzar un conocimiento "total" sobre todos y cada uno de sus ítems, aún cuando, inicie la navegación en un estado de desconocimiento absoluto.

La cuestión es ahora, ¿cuándo debe el sistema ejecutar dicho algoritmo para evaluar la consistencia de las reglas definidas sobre una estructura de aprendizaje? *Inicialmente*, cuando el autor crea una EC_A^i, el conjunto de reglas de conocimiento, $Rk(EC_A^i)$, es vacío y el conjunto de reglas de actualización, $Ru(EC_A^i)$, contiene las reglas generadas por defecto. En esta situación, obviamente *se satisface la consistencia del conjunto de reglas* $Rk(EC_A^i) \cup Ru(EC_A^i)$, ya que no existen restricciones que impidan el acceso a los ítems y puedan generar situaciones de inalcanzabilidad.

Mientras que el conjunto de reglas de conocimiento es vacío cualquier modificación en las reglas de actualización es indiferente para la consistencia del conjunto $Rk(EC_A^i) \cup Ru(EC_A^i)$. En este sentido, sería preferible que el autor modificase el conjunto de reglas de actualización por defecto antes de definir el conjunto de reglas de conocimiento. Aunque, por supuesto, puede modificar ambos conjuntos en el orden que desee y tantas veces como considere necesario.

Cuando el autor modifica el conjunto de reglas de conocimiento para expresar los requisitos pedagógicos existentes entre los ítems, puede originar sin darse cuenta alguna situación de interbloqueo entre dos o más ítems. Un interbloqueo hace inalcanzables a los ítems implicados, a no ser que la situación sea resuelta por las reglas de actualización. En este caso, es necesario *aplicar el algoritmo de comprobación de consistencia cuando el autor hace evolucionar el conjunto de reglas de conocimiento y/o de actualización*.

En principio, el algoritmo propuesto puede ser ejecutado por el sistema cada vez que el autor modifique, mediante la correspondiente acción evolutiva, el conjunto de reglas $Rk(EC_A^i) \cup Ru(EC_A^i)$. No obstante, se ha realizado un estudio (se detalla más adelante) acerca de cuándo es realmente necesaria esta comprobación y cuándo no. A partir de él, hemos podido concluir que existen modificaciones para las que no es preciso comprobar la consistencia del nuevo conjunto, $Rk'(EC_A^i) \cup Ru'(EC_A^i)$, si antes de llevar a cabo la modificación éste se encontraba en un estado consistente. Una vez identificados estos

cambios, el sistema ahorrará el esfuerzo necesario para ejecutar el algoritmo siempre que el cambio realizado lo permita.

En general una modificación llevada a cabo mediante una acción evolutiva, ACe, *no requiere la ejecución del algoritmo de comprobación de consistencia si el conjunto de reglas modificado es igual o menos restrictivo que el conjunto de reglas antes de la modificación*. Ver ecuación 1.

$$\text{ACe: } A \Rightarrow B, \text{ donde } A = Rk(EC_A^i) \cup Ru(EC_A^i) \text{ es consistente y} \quad (1)$$
$$B = Rk'(EC_A^i) \cup Ru'(EC_A^i) \text{ es igual o menos restrictivo que A.}$$

Diremos que **B es igual o menos restrictivo que A** si las condiciones de acceso en $Rk'(EC_A^i)$ son igual o menos restrictivas que en $Rk(EC_A^i)$ y las actualizaciones definidas en $Ru'(EC_A^i)$ son igual o mas generosas que las de $Ru(EC_A^i)$.

A continuación se enumeran y describen las modificaciones que no requieren recomprobar la consistencia del nuevo conjunto de reglas $Rk' \cup Ru'$ definido en EC_A^i. En la sección 4.1 se incluyen las modificaciones que afectan a Rk y en la sección 4.2 las que afectan a Ru. En todos los casos se indica la acción o acciones evolutivas implicadas y la transformación que tiene lugar tras su ejecución (véanse los capítulos 8 y 9). Además, para cada una se indica brevemente por qué el elemento modificado es menos o igual de restrictivo que el elemento inicial.

4.1 Modificaciones sobre Rk que no deben ser comprobadas

1) Se elimina la única regla de conocimiento asociada a un ítem i_j.

$ACe^2[Rk]$: $Rk(i_j) = \{Rk^1(i_j)\} \Rightarrow Rk'(i_j) = \varnothing$.

Rk' es menos restrictivo que Rk porque ahora el ítem i_j es inicialmente accesible. Rk' es igual de restrictivo que Rk si el antecedente de accesibilidad en $Rk^1(i_j)$ era vacío.

2) Se elimina una regla de conocimiento $Rk^m(i_j)$ asociada a un ítem i_j, y en el conjunto de reglas resultante existe al menos una con el antecedente de accesibilidad vacío.

$ACe^2[Rk] \vee ACe^6[Rk]$: $Rk(i_j) = \{Rk^1(i_j), ..., Rk^{m-1}(i_j), Rk^m(i_j), Rk^{m+1}(i_j), ..., Rk^n(i_j)\} \Rightarrow Rk'(i_j) = \{Rk^1(i_j), ..., Rk^{m-1}(i_j), Rk^{m+1}(i_j), ..., Rk^n(i_j)\}$

, donde $\exists_{i=1...n,\ i\neq m}$ tal que $Rk^i(i_j)$: true *and‴* $Antecedente\_Idoneidad^i \rightarrow Visitar(i_j)$.

Rk' es igual de restrictivo que Rk porque el ítem i_j sigue siendo inicialmente accesible a través de la regla $Rk^i(i_j)$ con antecedente de accesibilidad vacío.

3) Se añade una regla de conocimiento $Rk^m(i_j)$ con el antecedente de accesibilidad vacío al conjunto de reglas $Rk(i_j)$.

$ACe^1[Rk] \vee ACe^4[Rk] \vee ACe^6[Rk]$: $Rk(i_j)=\{Rk^1(i_j),...,Rk^{m-1}(i_j)\} \Rightarrow Rk'(i_j)=\{Rk^1(i_j),...,Rk^{m-1}(i_j),Rk^m(i_j)\}$

, donde $Rk^m(i_j)$: true *and‴* $Antecedente\_Idoneidad^m \rightarrow Visitar(i_j)$.

Rk' es menos restrictivo que Rk ya que, después de la modificación, el ítem i_j es inicialmente accesible a través de la nueva regla $Rk^m(i_j)$. Rk' es igual de restrictivo que Rk si éste contenía una o varias reglas con antecedente de accesibilidad vacío.

4) Se añade una regla de conocimiento $Rk^m(i_j)$ con antecedente de accesibilidad no vacío al conjunto de reglas de conocimiento $Rk(i_j)$, donde antes de la modificación existe al menos una regla de conocimiento. Esto es, $Rk(i_j) \neq \varnothing$.

$ACe^1[Rk] \vee ACe^4[Rk] \vee ACe^6[Rk]$: $Rk(i_j)=\{Rk^1(i_j),...,Rk^{m-1}(i_j)\} \Rightarrow Rk'(i_j)=\{Rk^1(i_j),..., Rk^{m-1}(i_j),Rk^m(i_j)\}$

, donde $Rk^m(i_j)$:Antecedente_Accesibilidad^m *and‴* Antecedente_Idoneidad$^m \rightarrow$ Visitar(i_j) y Antecedente_Accesibilidad$^m \neq$ true.

Si $\forall_{s=1..(m-1)}$ se cumple que $Rk^s(i_j)$:Antecedente_Accesibilidad^s *and‴* Antecedente_Idoneidad$^s \rightarrow$ Visitar(i_j) tiene Antecedente_Accesibilidad$^s \neq$ true, entonces Rk' es menos restrictivo que Rk ya que se ha añadido un camino más para hacer accesible a i_j.

Si por el contrario, $\exists_{s=1..(m-1)}$ tal que $Rk^s(i_j)$:Antecedente_Accesibilidad^s *and‴* Antecedente_Idoneidad$^s \rightarrow$ Visitar(i_j) con Antecedente_Accesibilidad^s = true, entonces Rk' es igual de restrictivo que Rk ya que i_j sigue siendo inicialmente accesible a través de la regla $Rk^s(i_j)$.

5) Se realiza cualquier modificación (añadir, eliminar o modificar restricciones de tipo Restricción^I) en el antecedente de idoneidad de una regla de conocimiento $Rk^m(i_j)$.

$ACe^3[Rk] \vee ACe^4[Rk] \vee ACe^5[Rk] \vee ACe^6[Rk]$: $Rk^m(i_j)$: Antecedente_Accesibilidad^m *and‴* Antecedente_Idoneidad$^m \rightarrow$ Visitar(i_j) $\Rightarrow$ Rk'$^m(i_j)$: Antecedente_Accesibilidad^m *and‴* Antecedente_Idoneidad'$^m \rightarrow$ Visitar(i_j).

Rk' es igual de restrictivo que Rk, ya que el antecedente de idoneidad no interviene en la accesibilidad del ítem.

6) Se elimina la restricción de accesibilidad definida sobre el ítem i_k en el antecedente de accesibilidad de la regla de conocimiento $Rk^m(i_j)$ asociada a un ítem i_j.

$ACe^3[Rk] \vee ACe^6[Rk]$: Rk'$^m(i_j)$: [Restricción$^A(i_1)$ ... *and* ~~Restricción$^A(i_k)$~~ ... *and* Restricción$^A(i_n)$]'^{m} *and‴* Antecedente_Idoneidad$^m \rightarrow$ Visitar(i_j).

Rk' es menos de restrictivo que Rk porque se ha quitado una restricción en uno de los caminos para acceder a i_j.

7) Se hace menos restrictiva la restricción de accesibilidad definida sobre el ítem i_k, esto es Restricción$^A(i_k)$, en el antecedente de accesibilidad de la regla de conocimiento $Rk^m(i_j)$.

$ACe^5[Rk] \vee ACe^6[Rk]$: $Rk^m(i_j)$: [Restricción$^A(i_1)$ ... and $K(i_k) \geq$ val and ... Restricción$^A(i_n)$]^m *and‴* Antecedente_Idoneidad$^m \rightarrow$ Visitar(i_j) $\Rightarrow$ Rk'$^m(i_j)$: [Restricción$^A(i_1)$ ... and $K(i_k) \geq$ val' and … Restricción$^A(i_n)$]'^{m} *and‴* Antecedente_Idoneidad$^m \rightarrow$ Visitar(i_j)

, donde se cumple que val' < val.

Rk' es menos de restrictivo que Rk porque se ha suavizado la restricción impuesta sobre un ítem en uno de los caminos de conocimiento para acceder a i_j.

8) Se añade una nueva restricción de accesibilidad o se hace más restrictiva una existente, Restricción$^A(i_k)$, en una regla Rk$^m(i_j)$, pero siguen existiendo otras reglas asociadas al ítem i_j con el antecedente de accesibilidad vacío.

ACe^4[Rk] $\vee$ ACe^5[Rk] $\vee$ ACe^6[Rk]: Rk'$^m(i_j)$: [Restricción$^A(i_1)$ ... and ... Restricción$^A(i_n)$ and Restricción$^{A'}(i_k)$]'^{m} *and‴* Antecedente_Idoneidad$^m \rightarrow$ Visitar(i_j)

, donde $\exists$ Rk$^i(i_j)$: true *and‴* Antecedente_Idoneidad$^i \rightarrow$ Visitar(i_j).

Rk' es igual de restrictivo que Rk porque aunque se haya hecho más restrictivo uno de los caminos para acceder a i_j, este ítem sigue siendo inicialmente accesible a través de la regla Rk$^i(i_j)$ cuyo antecedente de accesibilidad es vacío.

4.2 Modificaciones sobre Ru que no deben ser comprobadas

1) Se añade un predicado de actualización (de cualquier tipo) asociado a un ítem i_k en el cuerpo de la regla de actualización Ru(i_j).

ACe^3[Ru]: Ru(i_j): Si Visitado(i_j) entonces Actualización(i_j), ..., Actualización(i_m); $\Rightarrow$ Ru'(i_j): Si Visitado(i_j) entonces Actualización(i_j), ..., Actualización(i_m), Actualización(i_k);.

Ru' representa una actualización más generosa que Ru, ya que se ha incluido una nueva actualización tras la visita del ítem i_j.

2) En una regla Ru(i_j), un predicado de actualización Actualización$^a(i_k)$ es sustituido por otro predicado de actualización Actualización$^n(i_k)$ que consigue tras la visita repetida a i_j un grado de conocimiento sobre i_k mayor o igual que el conseguido por el antiguo.

ACe^5[Ru]: Ru(i_j): Si Visitado(i_j) entonces Actualización(i_j), ..., Actualización$^a(i_k)$, ...; $\Rightarrow$ Ru'(i_j): Si Visitado(i_j) entonces Actualización(i_j), ..., Actualización$^n(i_k)$, ...;

Podemos asegurar, sin ejecutar el algoritmo, que la nueva actualización Actualización$^n(i_k)$ no es menos generosa que la antigua Actualización$^a(i_k)$ si es posible afirmar, de acuerdo a lo especificado en la tabla 6 y sin evaluar ningún término del tipo $máximo_s(M_k[s,j])$ ó $máximo_s(M_k[s,k])$, que:

a) $M_k^n[j,k]$ es igual a 4 ó,

b) $M_k^a[j,k]$ es menor o igual que $M_k^n[j,k]$.

, donde $M_k^a[j,k]$ es el conocimiento máximo sobre i_k visitando i_j con los predicados Actualización(i_j) y Actualización$^a(i_k)$. Y $M_k^n[j,k]$ es el conocimiento máximo sobre i_k visitando i_j, con los predicados Actualización(i_j) y Actualización$^n(i_k)$.

Ejemplos de modificaciones que no necesitan volver a comprobar la consistencia son:

- Sustituir un predicado de tipo "primera-vez" por cualquier predicado de tipo "cada-vez" excepto el "relativo" con "decremento".

- Aumentar el grado de conocimiento implicado en una actualización "fija" y "absoluta".

- Aumentar el incremento o reducir el decremento en una actualización de tipo "incremental".

- Incluso disminuir el incremento en una actualización "incremental" y "absoluta" de tipo "cada-vez" ya que el conocimiento alcanzado va a seguir siendo "total", aunque ahora necesite de más visitas.

3) En el cuerpo de la regla Ru(i_j), el predicado de actualización sobre el ítem de la cabeza, Actualización(i_j), es modificado por otro predicado, Actualización'(i_j), en las formas que aquí se indican.

ACe[5][Ru]: Ru(i_j): Si Visitado(i_j) entonces Actualización(i_j), ...; $\Rightarrow$ Ru'(i_j): Si Visitado(i_j) entonces Actualización'(i_j), ...;, donde:

a) Actualización(i_j) = Inc-abs(i_j, val) $\Rightarrow$ Actualización'(i_j) = Fix-abs(i_j, "total").

b) Actualización(i_j) = Inc-abs(i_j, val1) $\Rightarrow$ Actualización'(i_j) = Inc-abs(i_j, val2), con val2 $\geq$ val1.

En ambos casos es indudable que la nueva actualización sobre i_j es más generosa que la antigua.

Si en el cuerpo de Ru(i_j) no existe ninguna actualización de tipo "primera-vez" y "relativa" también se incluyen las siguientes modificaciones:

c) Actualización(i_j) = Fix-abs(i_j, "total") $\Rightarrow$ Actualización'(i_j) = Inc-abs(i_j, val).

d) Actualización(i_j) = Inc-abs(i_j, val1) $\Rightarrow$ Actualización'(i_j) = Inc-abs(i_j, val2), con val2 < val1.

Obviamente en estos dos casos, el conocimiento obtenido sobre el ítem i_j tras visitarlo la primera vez es menor con el nuevo predicado que con el antiguo, lo cual afectaría negativamente a las actualizaciones "relativas" de tipo "primera-vez". Pero si no existen actualizaciones de este tipo, el conocimiento que se alcanza sobre i_j tras visitarlo repetidamente es igual que antes, esto es "total", con lo cual no hay que realizar de nuevo la comprobación de consistencia.

5. MODIFICACIÓN EN LAS ACCIONES EVOLUTIVAS PARA RK Y RU

En las secciones 4.1 y 4.2 se han expuesto los casos en los que no es necesario recomprobar la consistencia del conjunto de reglas de conocimiento y actualización tras un cambio en alguno de sus componentes. A continuación se realiza, en cierto modo, el

proceso complementario. Esto es, para cada acción evolutiva que lo requiera, se especifican las circunstancias en las que debe ejecutarse el algoritmo 2 de la sección 3, para determinar si tras la modificación realizada se sigue garantizando la alcanzabilidad de todos los ítems.

Por ejemplo, a partir de los puntos expuestos en la sección 4.1 se pueden deducir, entre otras muchas, las siguientes conclusiones:

A partir de 1) y 2) se infiere que la eliminación de una regla de conocimiento $Rk^m(i_j)$ sólo debe ser comprobada, si tras ello, el conjunto $Rk'(i_j)$ no queda vacío y ninguna de las reglas que contiene tiene el antecedente de accesibilidad *true*.

A partir de los puntos 3) y 4) se tiene que la adición de una regla de conocimiento $Rk^m(i_j)$ al conjunto $Rk(i_j)$ sólo debe ser comprobada cuando antes de la modificación $Rk(i_j)$ está vacío y la regla añadida $Rk^m(i_j)$ tiene al menos una restricción de accesibilidad.

A veces, las propiedades de los predicados de conocimiento (capítulo 8) permiten evitar la comprobación de consistencia tras una determinada modificación. Por ejemplo, si en una regla de conocimiento de autor, $Rk\text{-}autor^n(i_j)$, asociada a un ítem i_j se modifica un predicado OR_{SET} añadiendo dentro del conjunto un predicado más, esto supone añadir nuevas reglas en $Rk(i_j)$. Pero como el OR_{SET} tiene siempre al menos dos predicados, se garantiza que el conjunto $Rk(i_j)$ antes de la modificación no era vacío, por lo que, de acuerdo a lo anterior, no se requiere ninguna comprobación para las reglas añadidas.

Para averiguar si después del cambio propuesto, el nuevo conjunto de reglas de actualización y conocimiento se encuentra en un estado consistente, el algoritmo de comprobación debe ser ejecutado después de llevar a cabo la acción evolutiva. Por eso, la satisfacción de alcanzabilidad es una condición que debe evaluarse a posteriori, y es incluida como poscondición, en la tabla asociada a cada acción evolutiva.

Tabla 7: Poscondiciones de alcanzabilidad para las ACe[Rk]

ACe^1[Rk]	**AñadirRk(i_j, Rk-$autor^i(i_j)$): boolean;**
Poscondición	Si antes de realizar la modificación, el conjunto de reglas $Rk(i_j)$ es vacío, esto es: $Rk(i_j) = \varnothing$. Y tras ejecutar la modificación, ninguna de las reglas añadidas $\{Rk'^1(i_j), Rk'^2(i_j), ..., Rk'^m(i_j)\}$ tiene el antecedente de accesibilidad vacío, esto es: $Rk'^i(i_j) \notin Rk'(i_j)$ tal que $Rk'^i(i_j)$: true *and* ''' Antecedente_Idoneidad$^i \rightarrow$ Visitar(i_j). Es necesario validar la consistencia del conjunto $Rk' \cup Ru$ después de ejecutar la acción evolutiva y en caso de no satisfacerse deshacer y denegar el cambio.
ACe^2[Rk]	**EliminarRk(i_j, Rk-$autor^i(i_j)$): boolean;**
Poscondición	Si después de llevar a cabo la modificación, en el conjunto de reglas de conocimiento $Rk'(i_j)$ se cumple que:

	- $Rk'(i_j)$ no es vacío, esto es $Rk'(i_j) \neq \varnothing$ y - $Rk'(i_j)$ no contiene ninguna regla con antecedente de accesibilidad vacío, esto es: $Rk'^{i}(i_j) \notin Rk'(i_j)$ tal que $Rk'^{i}(i_j)$: true and''' $Antecedente\_Idoneidad^{i} \rightarrow Visitar(i_j)$. Es necesario validar la consistencia del conjunto $Rk' \cup Ru$ después de ejecutar la acción evolutiva y en caso de no satisfacerse deshacer el cambio.
ACe^4[Rk]	**AñadirPredicadoRk(i_j, Rk-autor^i(i_j), i_k, predicado(i_k)): boolean;**
Poscondición	Si la traducción interna del predicado(i_k) que se desea añadir implica una restricción de accesibilidad, $Restricción^A(i_k)$. Y después de ejecutar la acción evolutiva, en el nuevo conjunto de reglas asociadas al ítem i_j no existe ninguna regla con el antecedente de accesibilidad vacío, esto es: $Rk'^{i}(i_j) \notin Rk'(i_j)$ tal que $Rk'^{i}(i_j)$: true and''' $Antecedente\_Idoneidad^{i} \rightarrow Visitar(i_j)$. Es necesario validar la consistencia del conjunto $Rk' \cup Ru$ después de ejecutar la acción evolutiva y en caso de no satisfacerse deshacer y denegar el cambio.
ACe^5[Rk]	**ModificarPredicadoRk(i_j, Rk-autor^i(i_j), i_k, val1, val2): boolean;**
Poscondición	Si antes de ejecutar la acción evolutiva no existe ninguna regla de conocimiento en $Rk(i_j)$ con el antecedente de accesibilidad vacío, Y en la modificación propuesta: - El predicado que se desea modificar, predicado(i_k), es de tipo MayorE o MayorNE y el nuevo valor para instanciar el predicado, *val1*, es mayor que el que existe actualmente, ó - El predicado que se desea modificar, predicado(i_k), es de tipo IntervaloE o IntervaloNE y el nuevo valor para instanciar la segunda variable del predicado, *val2*, es mayor que el actual. Es necesario validar la consistencia del conjunto $Rk' \cup Ru$ después de ejecutar la acción evolutiva ya que las restricciones de accesibilidad de algunas reglas se han hecho más restrictivas. En caso de no garantizarse la alcanzabilidad de todos los ítems se deniega el cambio.
ACe^6[Rk]	**ModificarOR_{SET}PredicadoRk(i_j, Rk-autor^i(i_j), i_k, predicado^t(i_k), v1, v2): boolean;**
Poscondición	Se debe comprobar la consistencia del conjunto $Rk' \cup Ru$ en los siguientes casos: - Si se ha modificado el predicado^t(i_k) incluido en el predicado OR_{SET}, la comprobación se realiza en las mismas circunstancias explicadas para ACe^5[Rk]. - Si se ha eliminado el predicado^t(i_k) incluido en el OR_{SET}, la comprobación se realiza en las circunstancias explicadas para ACe^2[Rk] para cada una de las reglas eliminadas en $Rk(i_j)$ como consecuencia del cambio.

Tabla 8: Poscondiciones de alcanzabilidad para las ACe[Ru]

ACe^4[Ru]	**EliminarPredicado(i_j, i_k): boolean;**
Poscondición	Sea cual sea el predicado eliminado es necesario validar la consistencia del conjunto Rk ∪ Ru' después de ejecutar la acción evolutiva y en caso de no satisfacerse deshacer y denegar el cambio.
ACe^5[Ru]	**ModificarPredicado(i_j, i_k, predicado'(i_k)): boolean;**
Poscondición	Si el nuevo predicado propuesto para i_k, esto es, predicado'(i_k), supone una actualización menos generosa que el predicado(i_k) existente antes del cambio en Ru(i_j), es necesario validar la consistencia del conjunto Rk ∪ Ru' después de ejecutar la acción evolutiva y en caso de no satisfacerse deshacer y denegar el cambio. Para comparar la actualización propuesta en predicado'(i_k) con la de predicado(i_k) se sigue lo especificado en los puntos 2) y 3) de la sección 4.2.

6. COMPLEJIDAD DE LOS ALGORITMOS

Para concluir, se incluye en esta sección un análisis acerca de la complejidad temporal y espacial de cada uno de los dos algoritmos propuestos. En función de este análisis se presenta la posibilidad de hacer un uso combinado de ambos para evitar, siempre que sea posible, la ejecución del algoritmo de mayor costo. Se entiende por costo los recursos de espacio de almacenamiento y de, principalmente, tiempo de cómputo.

Es importante estudiar los costos de cómputo requeridos para la ejecución de los algoritmos que utilizamos para comprobar la consistencia del conjunto de reglas Rk ∪ Ru. Primero, porque es necesario verificar que es viable la aplicación de los mismos. Y segundo, y muy importante, para hacer notar que la complejidad obtenida mejora la complejidad exponencial que se obtendría si no tuviésemos otra forma para comprobar la alcanzabilidad de los ítems que generar y recorrer el espacio completo de estados posibles.

A continuación, el cálculo de complejidad computacional de cada algoritmo se realiza para el peor de los casos posibles, es decir el de mayor consumo. Éste depende, obviamente, del número de ítems mostrados en la presentación EC_P^j sobre la que se define la estructura conceptual de aprendizaje EC_A^i. Por lo que, el orden de complejidad es expresado en términos de *n*, siendo *n* el número de ítems indicado.

- Algoritmo para **comprobar la alcanzabilidad en Rk(EC_A^i)** (algoritmo 1, sección 2).

La *complejidad espacial* del algoritmo es de orden *O(n)*, ya que la ejecución de este algoritmo tan sólo requiere mantener en memoria un conjunto *A* que almacene los ítems accesibles hasta el momento.

La *complejidad temporal* del algoritmo es de orden $O(n^2)$ en el peor de los casos, ya que el número máximo de iteraciones del algoritmo es *n*. Situación que ocurre cuando en cada iteración del algoritmo un único ítem de los *n* existentes se hace accesible. Así, en

la iteración *i* del algoritmo es necesario evaluar la accesibilidad de *n-i* ítems, esto es, *n* en la iteración 0, *n-1* en la iteración 1, etc. Esto hace un total de $n + (n\text{-}1) + (n\text{-}2) + ... + 1$, esto es $(n^2 + n)/2$.

Este peor caso, únicamente tiene lugar cuando las reglas de conocimiento imponen un orden total en la visita de los *n* ítems de la EC_A^i. Este orden total hace que sólo sea posible una forma de recorrer los ítems mostrados, lo cual merma enormemente las ventajas y posibilidades del sistema hipermedia y por tanto cabe esperar que sea raro que ocurra.

- Algoritmo para **comprobar la alcanzabilidad en Rk ∪ Ru** (algoritmo 2, sección 3).

La *complejidad espacial* de este algoritmo es de orden $O(n^2)$. Esto se debe a que la cantidad de memoria necesaria durante la ejecución del algoritmo viene determinada por la dimensión $n \times n$ de la matriz M_k, que en cada momento almacena el conocimiento máximo que es posible alcanzar sobre un ítem i_k visitando un ítem accesible i_j.

La *complejidad temporal* para este algoritmo es de orden $O(n^3)$ en el peor caso. Las circunstancias supuestas para el peor caso son las siguientes:

1. De nuevo, los *n* ítems se van haciendo accesibles uno a uno, por lo tanto, es necesario ejecutar *n* iteraciones completas del algoritmo. En este caso la mayor complejidad no viene determinada por la evaluación de las reglas $Rk(i_j)$ asociadas a los *n-i* ítems no accesibles en la iteración *i*, sino por el cálculo de la matriz M_k necesaria para realizar dicha evaluación.

2. En cada iteración hay que calcular las *n* celdas de una fila de la matriz M_k. Concretamente la fila calculada es la que corresponde el ítem considerado. Esto es, siendo i_j el último ítem en hacerse accesible ($A_N = \{i_j\}$), es necesario calcular para k = 1..*n* la celda M_k [j, k]. Esta situación de peor caso sólo sucede si todas las reglas de actualización incluyen en su cuerpo un predicado de actualización para cada ítem incluido en la estructura de navegación.

3. En la evaluación de cada celda M_k [j,k] hay que resolver un término de tipo $máximo_s(M_k\ [s,j])$ o $máximo_s(M_k\ [s,k])$. Para resolver este término es necesario calcular el máximo de *n* celdas de la matriz M_k. Esta situación de peor caso ocurre si la actualización del ítem i_k en el cuerpo de la regla i_j es de tipo $Fix\text{-}rel_{first}$, $Inc\text{-}rel_{first}$ o $Inc\text{-}abs_{first}$.

A partir de 1, 2 y 3, el orden de complejidad temporal que se obtiene para el peor caso con este algoritmo es $n \times n \times n$, esto es $O(n^3)$.

Ambos algoritmos tienen **complejidad polinomial**, de manera que su ejecución, tras las modificaciones que así lo requieran, no implica un esfuerzo excesivo. Sobre todo si tenemos en cuenta que:

a) la complejidad de un caso medio va a ser mucho menor que la del peor caso considerado (realmente improbable),

b) el número de ítems por presentación no debe ser muy alto, normalmente entre 10 y 20, y

c) los cálculos implicados son muy sencillos y por lo tanto, serán realizados a gran velocidad. Piense por ejemplo, en el tiempo que emplea un procesador actual para hacer una comparación de dos valores numéricos, efectuar una suma, o resolver un máximo de unos cuantos valores.

La complejidad del primer algoritmo es mejor que la complejidad del segundo, puesto que el exponente al que se eleva *n* en el primer caso es 2 mientras que en el segundo caso es 3. Por ello, preferimos la ejecución del primer algoritmo siempre que sea posible asegurar la alcanzabilidad de todos los ítems, simplemente evaluando las reglas de conocimiento. Reservamos la ejecución del segundo algoritmo para aquellos casos en que el primero no es suficiente, esto es, cuando existen interbloqueos en las restricciones de accesibilidad de las reglas de conocimiento y éstos pueden ser resueltos o no por las reglas de actualización.

- Comprobar la **alcanzabilidad en Rk ∪ Ru empleando fuerza bruta**.

Determinar por fuerza bruta la alcanzabilidad de todos los ítems de una estructura conceptual, supone generar a partir de un estado inicial de desconocimiento absoluto todos los caminos de navegación posibles. De forma que, sólo después de haber analizando todas las posibilidades es correcto asegurar la ausencia de un recorrido a través del cuál es imposible acceder a un determinado ítem o conjunto de ítems.

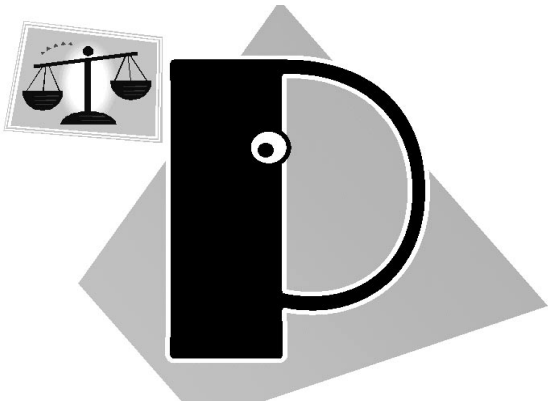
Sin embargo, está ampliamente demostrado que la generación de dicho espacio de estados, tiene un orden de complejidad temporal y espacial $O(m^s)$, donde *s* es el número de elementos incluido en cada estado y *m* el número de posibles valores para cada elemento del estado.

En nuestro caso un estado de conocimiento (Def 8.2 en capítulo8), contiene el grado de conocimiento que el usuario posee acerca de cada ítem. Con lo que, el número de elementos que forman cada estado coincide con el número de ítems en la estructura de navegación, o sea *n*. El número de posibles valores para cada elemento es el número de etiquetas semánticas diferentes para representar un grado de conocimiento, 5 en nuestro caso. Resultando la complejidad por fuerza bruta de orden $O(5^n)$ mucho peor que la obtenida para los algoritmos 1 y 2.

Queda así patente la eficiencia de los algoritmos propuestos para garantizar la consistencia del conjunto de reglas Rk ∪ Ru, sin necesidad de generar todo el espacio de estados, lo cual supondría una complejidad exponencial intratable. La consistencia de Rk ∪ Ru implica que todos los ítems son alcanzables sin importar el recorrido realizado hasta el momento, esto constituye una propiedad fundamental durante la navegación del usuario y como se verá más adelante para la generación de rutas guiadas (capítulo 20).

CAPÍTULO 11

Reglas de Peso



Resumen

Se necesita un mecanismo que permita obtener el conocimiento del usuario acerca de un concepto a partir del conocimiento que éste posee sobre los ítems de información que lo tratan. Este mecanismo se establece sobre un conjunto de reglas de peso, especificadas formalmente en el presente capítulo. Se establece la representación de las reglas de peso, su utilidad y la forma en que se ejecutan. Se describe el modo en que el sistema genera las reglas de peso por defecto y cómo el autor puede modificar éstas para establecer a su criterio qué ítems son más relevantes para comprender un concepto. Por último, se detalla un conjunto de acciones evolutivas, ejecutadas por el sistema y por el autor, para crear y actualizar el conjunto de reglas de peso definido sobre una estructura conceptual de memorización.

Tabla de contenidos

Reglas de Peso

1. INTRODUCCIÓN

Las reglas de peso determinan el *conocimiento del usuario sobre los conceptos* del sistema hipermedia basándose en el grado de conocimiento que éste posee acerca de los ítems de información que describen, explican, ejemplifican o en definitiva tratan sobre ellos.

A diferencia de las reglas de conocimiento y actualización, las reglas de peso se definen sobre la propia estructura de memorización y no en cada una de sus presentaciones. Concretamente, para cada concepto incluido en la estructura conceptual de memorización existe una única regla de peso. De forma abreviada notamos **$Rw(EC_M)$** al conjunto de reglas de peso definidas para la estructura conceptual de memorización EC_M. Y fijada la estructura de memorización, identificamos la regla de peso de un concepto c_k como **$Rw\ (c_k)$**.

> **Def 11.1 [Regla de peso]** La regla de peso $Rw(c_k)$ ligada a un concepto c_k, calcula el grado de conocimiento del usuario sobre dicho concepto a partir del grado de conocimiento que éste posee sobre cada uno de los ítems asociados funcionalmente al concepto en la EC_M. Obviamente, el contenido de dos ítems asociados a un mismo concepto no tiene porqué ser de igual relevancia para el conocimiento de éste. Por esta razón, en la regla de peso se pondera la importancia de cada ítem implicado en el cálculo del conocimiento sobre el concepto.

El conjunto de reglas de peso es, por tanto, el mismo en todas las estructuras conceptuales de aprendizaje, EC_A^i, que se definen sobre una misma estructura de memorización EC_M. Esto permite que cuando un usuario pasa de usar una estructura de aprendizaje a otra definida para la misma EC_M, su conocimiento sobre los conceptos sea calculado de forma idéntica, no cambiando hasta que, como consecuencia de la navegación realizada en la nueva estructura, aumente su conocimiento sobre algunos ítems. No tendría sentido que al cambiar de estructura conceptual de aprendizaje, el conocimiento del usuario sobre un concepto c_k fuese distinto a pesar de ser idéntico su conocimiento acerca de todos los ítems asociados a éste.

La regla de peso de un concepto involucra el grado de conocimiento que el usuario posee sobre todos los ítems asociados a él en la estructura conceptual de memorización. Por lo tanto, si en la presentación sobre la que se define una EC_A^i no aparecen todos los ítems asociados a un concepto c_k, un usuario que navega únicamente en esa estructura no puede lograr un conocimiento "total" sobre c_k, aunque alcance conocimiento "total" sobre todos los ítems asociados a él en la estructura actual.

La anterior situación se debe a que el usuario no tiene forma de incrementar su conocimiento sobre los ítems asociados a c_k que quedan fuera de la estructura actual. Así, si el usuario está interesado en aprender más sobre ese concepto debe cambiar a otra estructura conceptual de aprendizaje EC_A^j que le permita tener acceso al resto de ítems asociados a c_k. En general, para obtener conocimiento "total" sobre un concepto, es necesario conocer perfectamente todos sus ítems, lo que puede hacerse navegando

una o varias estructuras que en conjunto muestren todos los ítems asociados al concepto.

2. REPRESENTACIÓN DE LAS REGLAS DE PESO

En las reglas de peso, para nombrar a un ítem, usamos la notación $c_k.i_j$ donde c_k es el nombre del concepto al que se asocia el ítem e i_j su identificador unívoco. De este modo, un ítem i_j que está asociado a varios conceptos c_1, c_2,..., c_n será notado como $c_1.i_j$, $c_2.i_j$, ..., $c_n.i_j$ en las respectivas reglas de peso $Rw(c_1)$, $Rw(c_2)$, ..., $Rw(c_n)$, aunque en todos los casos se trate del mismo ítem. Esta notación no tiene otro fin que hacer explícita la asociación del ítem con el concepto de la regla de peso. Y en cualquier caso podría ser sustituida simplemente por i_j.

$$Rw(c_k): K(c_k) = \sum_{j=1}^{n} w_j \times K(c_k.i_j), \text{ con } i_1, i_2, \ldots, i_n \text{ los ítems asociados a } c_k \quad (1)$$

Como puede observarse en la ecuación 1, las reglas de peso son representadas mediante formulas matemáticas sencillas que calculan el grado de conocimiento del usuario sobre un concepto "combinando" su grado de conocimiento sobre los ítems asociados a éste. Por ello, la regla de peso $Rw(c_k)$ tiene un término para cada ítem i_j asociado al concepto c_k en la EC_M, esto es $\forall i_j$ tal que $<c_k, r_f, i_j> \in Af(EC_M)$.

Este término no es otro que el grado de conocimiento poseído sobre el ítem $c_k.i_j$, y es expresado mediante la función de conocimiento $K(c_k.i_j)$. Esta misma función se utiliza para representar el grado de conocimiento sobre el concepto c_k, esto es $K(c_k)$. Puesto que el conocimiento sobre un ítem i_j es el mismo independientemente del concepto al que se asocie, todos los términos $K(c_1.i_j)$, $K(c_2.i_j)$, ..., $K(c_n.i_j)$ tienen idéntico valor en las diferentes reglas de conocimiento donde aparecen. Esto es, $K(c_1.i_j) = K(c_2.i_j) = ... = K(c_n.i_j) = K(i_j)$. Recuérdese que no es necesario hacer distinción del concepto al que se asocia un ítem para definir la adquisición de conocimiento tras su lectura en las reglas de actualización (capítulo 9) o los requisitos de conocimiento necesarios para comprenderlo en las reglas de conocimiento (capítulo 8).

En cada regla de peso $Rw(c_k)$, el término $K(c_k.i_j)$ referente a un ítem i_j es ponderado mediante un factor multiplicador denominado w_j. Este factor representa el peso que el conocimiento sobre ese ítem, $K(c_k.i_j)$, tiene a la hora de calcular el conocimiento acerca del concepto c_k. De esta forma, aunque el valor del término asociado a un ítem i_j sea igual en todas las reglas de peso en las que aparece, seguramente será distinto el factor multiplicador que lo acompaña. Esto es necesario, ya que el contenido del ítem i_j puede ser muy importante para comprender el concepto c_1, pero poco relevante en el aprendizaje del concepto c_2 sobre el que también trata. Por lo tanto el peso w_j podrá ser diferente para los términos $K(c_1.i_j)$, $K(c_2.i_j)$, ..., $K(c_n.i_j)$ en las reglas de peso $Rw(c_1)$, $Rw(c_2)$, ..., $Rw(c_n)$.

Los pesos w_j involucrados en una regla de peso constituyen una distribución de porcentajes y como tal debe satisfacer las restricciones semánticas impuestas en la ecuación 2.

$$\forall j = 1..n : 0 \leq w_j \leq 1 \text{ y } \sum_{j=1}^{n} w_j = 1 \qquad (2)$$

Estas restricciones obligan, por un lado, a que el valor de un peso w_j sea un número real positivo comprendido en el intervalo [0, 1]. Y por otro, a que la suma de todos los pesos incluidos en una regla de peso de como resultado 1.

Concretamente, la última restricción es necesaria para evitar las siguientes situaciones no deseables:

a) El conocimiento sobre c_k obtenido al ejecutar su regla de peso es mayor que el valor de conocimiento máximo[1].

Si $\sum_{j=1}^{n} w_j > 1$ entonces Rw(c_k) puede generar K(c_k) > 4

Por ejemplo dada la regla de peso Rw(c1) : K(c1) = ½ K(c1.i1) + ½ K(c1.i2) + ½ K(c1.i3) cuya distribución de pesos suma 1,5, se obtiene un valor de conocimiento 6 para el concepto c1 si el valor de conocimiento de sus tres términos es el máximo (4).

b) El conocimiento sobre c_k obtenido al ejecutar su regla de peso no es "total" a pesar de que el grado de conocimiento poseído sobre cada uno de sus ítems asociados es "total".

Si $\sum_{j=1}^{n} w_j < 1$ entonces Rw(c_k) genera K(c_k) < 4

Ahora, dada la regla de peso Rw(c1) : K(c1) = ¼ K(c1.i1) + ¼ K(c1.i2) + ¼ K(c1.i3) cuya distribución de pesos suma 0,75, se obtiene un valor de conocimiento 3 para el concepto c1 cuando el valor de conocimiento de sus tres términos es 4.

Si además se quisiera asegurar que la única forma de obtener conocimiento "total" sobre un concepto c_k es conocer, con ese mismo grado, todos los ítems asociados a él, sería necesario exigir que el peso w_j fuese distinto de cero para todo término K(c_k.i_j) de la regla. Ya que en otro caso, podría ocurrir que:

c) El conocimiento sobre c_k obtenido al ejecutar su regla de peso sea "total" a pesar de que existen uno o varios ítems asociados a éste cuyo grado de conocimiento es menor que "total".

Si $\exists w_j = 0$ entonces Rw(c_k) puede generar K(c_k) = 4 con K(c_k.i_j) < 4

[1] De nuevo volvemos a considerar la existencia de cinco etiquetas semánticas, por lo que el valor de conocimiento máximo es 4 que corresponde a la etiqueta "total".

No obstante, nosotros consideramos que es conveniente dejar libertad al autor para ignorar un determinado ítem en el cálculo del conocimiento del concepto al que se asocia, aunque seguramente esto no sea lo habitual. Por este motivo decidimos no exigir la condición c) durante la definición de los pesos de una $Rw(c_k)$.

Cuando todos los pesos w_j incluidos en la regla $Rw(c_k)$ son idénticos, es decir, se trata de una distribución de porcentajes uniforme, el conocimiento acerca del concepto $c_{k,}$ se obtiene como el conocimiento medio de todos los ítems asociados. En otro caso, un peso w_j más alto que otro w_m hace que el conocimiento acerca del ítem i_j sea más significativo que el conocimiento sobre i_m en el cálculo de $K(c_k)$.

3. APLICACIÓN DE LAS REGLAS DE PESO

Las reglas de peso son ejecutadas automáticamente por el sistema para calcular el conocimiento del usuario sobre los conceptos ofrecidos por el sistema hipermedia. La ejecución de una regla de peso $Rw(c_k)$ asociada al concepto c_k debe ser llevada a cabo cada vez que el grado de conocimiento del usuario sobre alguno de los ítems i_j asociados a dicho concepto se incremente.

3.1 Utilidad de las reglas de peso

Ahora bien, ¿por qué es interesante conocer el grado de conocimiento que el usuario posee sobre los conceptos?, ¿cuál es su utilidad? Bueno, tal y como se explica más adelante, calcular el conocimiento del usuario sobre los conceptos del sistema hipermedia, tiene una doble utilidad.

Por un lado, desde un *punto de vista pedagógico* es muy importante que el usuario sea consciente de con qué nivel de detalle, es decir, cuánto conoce los distintos conceptos del sistema hipermedia. Esto se debe a que es más fácil para el usuario evaluar su proceso de aprendizaje a partir del conocimiento conseguido acerca de los conceptos explicados que de los trozos de información puntuales que los explican.

Es decir, está bien saber cuál es el conocimiento que se posee sobre el contenido de un determinado documento, pero al final lo realmente importante es saber si se ha aprendido o no el concepto o conceptos que se trataba de transmitir. Para ello, se informa sobre el conocimiento de los conceptos en la estructura de navegación, mejorando así la interacción del usuario con el sistema.

Por otro lado, esta información puede ser utilizada para *inferir la experiencia* media del usuario *en el tema* tratado por el sistema hipermedia. Este dato es utilizado, junto con otra información del usuario, para determinar, por ejemplo, la estructura conceptual de aprendizaje que mejor se ajusta a su perfil (capítulo 14).

3.2 Ejecución de una regla de peso

La regla de peso (ecuación 1) asociada a un concepto c_k es una sumatoria de *n* términos, cada uno de los cuales supone la multiplicación de dos valores, uno real, w_j, y otro entero, $K(c_k.i_j)$. Es por tanto una fórmula matemática donde tanto los operandos como el resultado son valores numéricos.

Para ejecutar una regla de peso $Rw(c_k)$, el sistema emplea los valores numéricos entre 0 y 4 equivalentes a las etiquetas semánticas que representan el grado de conocimiento sobre cada ítem implicado, esto es $K(c_k.i_j)$. El resultado numérico obtenido tras realizar las operaciones pertinentes, representa el grado de conocimiento sobre el concepto c_k. Sin embargo, debido a que el factor multiplicador es un número real, el resultado de ejecutar la regla de peso Rw(ck) es también un número real, donde a menudo la parte decimal es distinta de 0.

El problema es que, tal y como está definida, la función de conocimiento $K(c_k)$ sólo permite cinco valores distintos, 0, 1, 2, 3 y 4, todos ellos enteros. Para solucionarlo se aproxima al más cercano de estos valores el resultado obtenido ejecutando la regla de peso. Esto es posible, gracias a que, al exigir una distribución de pesos igual a 1, está garantizado que el resultado de la regla es un número comprendido entre 0 y 4.

El procedimiento de transformación es muy sencillo. Basta con aplicar una regla de redondeo muy básica: "si la parte decimal del valor numérico obtenido es mayor que 0,5 se redondea a la alza, esto es se incrementa en 1 la parte entera, y si es menor que 0,5 se redondea a la baja, esto es se desecha la parte decimal".

El único caso, donde cabe duda en la transformación propuesta tiene lugar cuando la parte decimal es exactamente 0,5. Por ello, el esquema de transformación propuesto (véase la tabla 1) presenta dos variantes, una por defecto y otra por exceso, para el mismo esquema.

Respecto a la notación utilizada:

Definimos *[v1, v2]* como un intervalo cerrado, donde ambos valores *v1* y *v2* están contenidos. Esto es, $s \in [v1, v2]$ si $v1 \leq s \leq v2$.

Definimos *(v1, v2]* como un intervalo abierto por la izquierda donde el valor *v1* no está incluido. Esto es, $s \in [v1, v2]$ si $v1 < s \leq v2$.

Definimos *[v1, v2)* como un intervalo abierto por la derecha donde el valor *v2* no está incluido. Esto es, $s \in [v1, v2]$ si $v1 \leq s < v2$.

Tabla 1: Transformación del resultado a un valor entero

Resultado numérico de $Rw(c_k)$		Valor entero para $K(c_k)$
Esquema **Por defecto**	Esquema **Por exceso**	
[0 – 0,5]	[0 – 0,5)	0 ("nulo")
(0,5 – 1,5]	[0,5 – 1,5)	1 ("bajo")
(1,5 – 2,5]	[1,5 – 2,5)	2 ("medio")
(2,5 – 3,5]	[2,5 – 3,5)	3 ("alto")
(3,5 – 4]	[3,5 – 4]	4 ("total")

Cuando el resultado de la regla es un número real que se queda justo en la mitad de dos grados de conocimiento, supongamos 1,5, el sistema de transformación propuesto puede elegir por exceso el grado superior, 2 en este caso, o por defecto el grado inferior, 1 en este caso.

Dicha elección no tiene demasiada trascendencia. De todos modos, será el autor el que decida el método que desea aplicar (por exceso o por defecto), pudiendo cambiar esta decisión cada vez que lo considere necesario. En caso de que el autor no tome ninguna decisión al respecto, el sistema aplica el esquema de transformación por defecto.

Una vez realizada la transformación, el conocimiento acerca del concepto es mostrado al usuario a través de una etiqueta semántica, justamente la equivalente al valor numérico obtenido. Cuando en la transformación se pierde precisión, es decir, el valor de conocimiento real del concepto es menor o mayor que el asignado tras el redondeo, es necesario informar al usuario de que el grado de conocimiento indicado en la etiqueta es aproximado. Esto permite dejar claro al usuario que tener un conocimiento aproximadamente "total"(≈ "total") sobre el concepto c_k no es lo mismo que conocerlo completamente, aunque se esté cerca de ello (capítulo 18, Navegación por conocimiento).

4. ¿Quién Define las Reglas de Peso?

4.1 Reglas de peso por defecto

Inicialmente las reglas de peso son generadas de forma automática por el propio sistema. El procedimiento ejecutado para ello es bien sencillo. Para cada concepto c_k de la estructura conceptual de memorización, el sistema genera una regla de peso, $Rw(c_k)$, en la cuál todos los ítems i_j asociados funcionalmente a c_k tienen el mismo peso w_j.

Para obtener este peso por defecto se divide la unidad entre el número de ítems implicados en la regla. Es decir si en la EC_M hay *n* ítems asociados al concepto c_k, el peso de cada término $K(c_k.i_j)$ es 1/n. Las ecuaciones 3, 4 y 5 muestran, en distintos formatos (de menos a más simplificados), la regla de peso generada por defecto para un concepto c_k. En todas ellas se considera la existencia de *n* ítems, i_1, i_2,..., i_n, asociados a c_k.

$$Rw(c_k): K(c_k) = \frac{1}{n} \times K(c_k.i_1) + \frac{1}{n} \times K(c_k.i_2) + ... + \frac{1}{n} \times K(c_k.i_n) \qquad (3)$$

$$Rw(c_k): K(c_k) = \sum_{j=1}^{n} \frac{1}{n} \times K(c_k.i_j) \qquad (4)$$

$$Rw(c_k): K(c_k) = \frac{1}{n} \times \sum_{j=1}^{n} K(c_k.i_j) \qquad (5)$$

En la última fórmula, la más simplificada, puede observarse más claramente cómo por defecto el conocimiento que tiene el usuario sobre un concepto es el conocimiento medio que dicho usuario posee sobre los ítems que tratan el concepto.

4.2 Modificación de las reglas de peso

El sistema es capaz de definir un conjunto de reglas de peso por defecto para una EC_M. Esto permite descargar al autor de tal tarea pero, por supuesto, éste puede modificar la ponderación de cualquier regla de peso siempre que lo considere necesario. Permitiéndole así, reflejar la desigual importancia que el conocimiento de cada ítem juega en el proceso de aprendizaje del concepto.

La modificación de los pesos de una regla debe ser realizada por el autor a través de la ejecución de la acción evolutiva correspondiente. Por supuesto, esta acción se encarga de garantizar el cumplimiento de las restricciones exigidas para la distribución de pesos en la ecuación 2, denegando el cambio en caso de no satisfacerse.

Debido a su sencillez, la fórmula matemática utilizada para expresar una regla de peso es perfectamente comprensible para la mayoría de los autores. Sin embargo, proponemos una interfaz gráfica, todavía más intuitiva y cómoda, que permite definir los pesos de la regla sin necesidad de visualizar la fórmula donde se utilizan.

En la figura 1 se muestra la interfaz propuesta para establecer la regla de peso $Rw(c_k)$. En ella, se exhibe gráficamente el concepto c_k con todos los ítems i_j asociados a éste en la EC_M. Junto a cada ítem i_j aparece una especie de folio donde el autor debe escribir la relevancia que su contenido juega en la comprensión del concepto. Esa importancia es el peso w_j asociado al ítem i_j, y puede ser expresado como un porcentaje, ya que a menudo, éste es más intuitivo que el número real entre 0 y 1 que le corresponde.

De esta forma el autor únicamente se preocupa de establecer el porcentaje en el que, el conocimiento de cada ítem repercute sobre el conocimiento del concepto central. Esto permite que definir las reglas de peso sea una tarea tan fácil como rellenar un sencillo formulario.

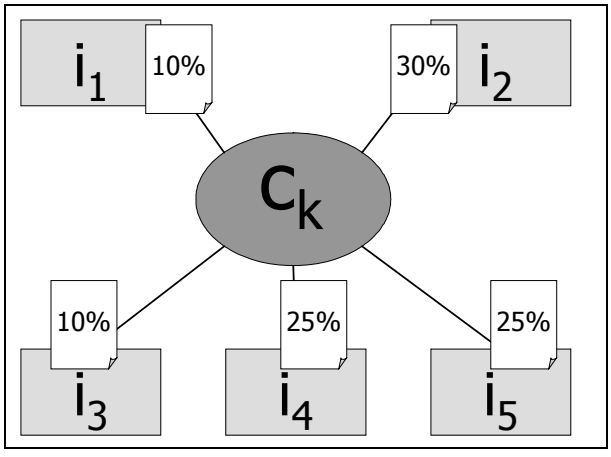


Figura 1: Interfaz para $Rw(c_k)$

De acuerdo a lo especificado en la ecuación 1, la regla de peso, $Rw(c_k)$, establecida por el autor para el concepto c_k en la figura 1 es la siguiente:

$$K(c_k) = 0{,}1 \times K(c_k.i_1) + 0{,}3 \times K(c_k.i_2) + 0{,}1 \times K(c_k.i_3) + 0{,}25 \times K(c_k.i_4) + 0{,}25 \times K(c_k.i_5)$$

Si suponemos un usuario con el estado de conocimiento parcial que se refleja en la tabla 2:

Tabla 2: Estado de conocimiento del usuario

K(i1)	K(i2)	K(i3)	K(i4)	K(i5)	...
1	1	2	1	3	...

El resultado de aplicar la regla $Rw(c_k)$ es 1,6. Aplicando el esquema de transformación de la tabla 1, el grado de conocimiento que nuestro usuario posee sobre el concepto c_k, es $K(c_k) \approx 2$.

5. ACCIONES EVOLUTIVAS DE LAS REGLAS DE PESO

En la sección anterior se ha descrito cómo se generan de forma automática un conjunto de reglas de peso que después pueden ser modificadas por el autor del sistema hipermedia. Estas modificaciones obedecen a dos motivos diferentes:

a) El autor decide cambiar la distribución de pesos de una regla o la variante del esquema de transformación aplicado (de por exceso a por defecto o viceversa), o

b) El autor realiza cambios en determinados elementos del Sistema de Memorización que deben ser propagados a las reglas de peso, por ejemplo elimina la asociación entre un ítem y un concepto o el propio ítem. Dicha propagación es realizada por el sistema, de forma que este tipo de cambios no son ejecutados directamente por el autor, sino como consecuencia de otros cambios que éste realiza.

Por tanto, el autor ejecuta modificaciones sobre las reglas de peso tanto de forma directa (a), como de forma indirecta (b).

Respecto a los pesos, la distribución de pesos en una regla $Rw(c_k)$ puede ser:

- la distribución uniforme asignada por defecto, o
- una distribución establecida por el propio autor.

En cualquier momento el autor puede modificar la distribución de pesos por defecto de una regla utilizando la acción evolutiva “CambiarPesosRw”. Pero también se le permite el paso inverso. Esto es, usando la acción evolutiva “InicializarPesosRw” puede reestablecer los pesos de una regla a la distribución uniforme por defecto.

Conocer el tipo de distribución que existe en una regla es importante. Ya que, cuando la distribución de pesos es por defecto, al añadir un nuevo término en una regla, el sistema es capaz de recalcular los pesos para que siga manteniéndose uniforme. Cuando la distribución es de autor, el sistema no tiene forma de saber cómo ni cuánto la adición del nuevo término modifica los pesos existentes. Por eso, lo más sensato es asignar peso

0 al nuevo término. Y sugerirle al autor que actualice la distribución de pesos, si lo considera oportuno.

Por ello, asociada a cada nueva regla de peso $Rw(c_k)$, se crea y mantiene una variable denominada *DistribuciónPesosRw(c_k)*, cuyo valor permite distinguir inmediatamente si los pesos de la regla $Rw(c_k)$ han sido definidos por el autor, por el sistema o se trata de una regla sin ningún término. Concretamente, la interpretación de esta variable es la siguiente:

- DistribuciónPesosRw(c_k) = "autor", indica que la distribución de pesos de la regla $Rw(c_k)$ ha sido establecida por el autor.

- DistribuciónPesosRw(c_k) = "sistema", indica que la distribución de pesos de la regla $Rw(c_k)$ es la distribución de pesos uniforme definida por el sistema.

- DistribuciónPesosRw(c_k) = "ninguna", indica que la distribución de pesos de la regla $Rw(c_k)$ es vacía, ya que la regla no tiene ningún término.

A continuación, se enumeran las acciones evolutivas disponibles para modificar las reglas de peso asociadas a una EC_M. De nuevo, en cada acción, se especifica: argumentos, definición, precondición, efecto, salida, y ejecutada_por (autor, sistema o ambos). Puesto que no se permiten dos conceptos con idéntico nombre en una misma EC_M, el nombre de un concepto se utiliza como su identificador.

5.1 Crear una nueva regla de peso

Tabla 3: Acciones evolutivas para modificar las reglas de peso –CrearRw–

ACe[1] [Rw]	**CrearRw (c_k)**
Argumentos	**c_k** es el nombre del concepto para el que se va a crear la regla de peso.
Definición	Esta acción evolutiva crea una regla de peso, $Rw(c_k)$, para el nuevo concepto c_k. La regla creada fija a "nulo" el grado de conocimiento sobre el concepto c_k, ya que aún no tiene ningún ítem asociado.
Precondición	El concepto c_k existe en la estructura conceptual de memorización actual. Esto es, $c_k \in C(EC_M)$.
	No existe ninguna regla de peso asociada al concepto c_k en la EC_M actual. Esto es, $Rw(c_k) \notin Rw(EC_M)$.
Efecto	La regla de peso creada para el concepto c_k queda como sigue: $Rw(c_k) : K(c_k) = 0$. La nueva regla de peso $Rw(c_k)$ se añade al conjunto de reglas de peso existentes. Esto es: $Rw'(EC_M) = Rw(EC_M) \cup \{Rw(c_k)\}$. Se crea una variable denominada DistribuciónPesosRw(c_k), cuyo valor inicial es "ninguna", indicando que en la regla $Rw(c_k)$ no se ha definido ningún peso (ni por parte del sistema ni del autor). Esto es: DistribuciónPesosRw(c_k) = "ninguna".

Salida	No es necesaria ninguna salida de control. Siempre se cumplen las precondiciones porque esta acción evolutiva es ejecutada únicamente por el sistema cuando se crea un nuevo concepto c_k en la EC_M actual.
Ejecutada_por	Sistema.

5.2 Eliminar una regla de peso

Tabla 4: Acciones evolutivas para modificar las reglas de peso –EliminarRw–

ACe^2 [Rw]	**EliminarRw(c_k)**
Argumentos	**c_k** es el concepto cuya regla de peso se desea eliminar.
Definición	Se elimina del conjunto $Rw(EC_M)$ la regla de peso asociada al concepto c_k, esto es $Rw(c_k)$.
Precondición	El concepto c_k no existe en la estructura conceptual de memorización actual. Esto es, $c_k \notin C(EC_M)$.
	En la estructura conceptual de memorización actual existe una regla de peso asociada al concepto c_k. Esto es, $Rw(c_k) \in Rw(EC_M)$.
Efecto	La regla de peso $Rw(c_k)$ desaparece del conjunto de reglas de peso asociadas a la estructura conceptual de memorización (obviamente también se destruye la variable DistribuciónPesosRw(c_k) asociada). $Rw'(EC_M) = Rw(EC_M) – Rw(c_k)$.
Salida	No es necesaria ninguna salida de control, ya que esta acción sólo es ejecutada por el sistema cuando el concepto c_k es eliminado de la EC_M.
Ejecutada_por	Sistema.

5.3 Añadir un nuevo término a una regla de peso

Tabla 5: Acciones evolutivas para modificar las reglas de peso –AñadirTérminoRw–

ACe^3 [Rw]	**AñadirTérminoRw(c_k, i_j)**
Argumentos	**c_k** es el nombre del concepto en cuya regla de peso se desea añadir un nuevo término. **i_j** es el identificador del nuevo ítem asociado al concepto c_k para el que se va a añadir el término.
Definición	Esta acción evolutiva añade un nuevo término $K(c_k.i_j)$ a la regla de peso $Rw(c_k)$. El peso w_j asignado al nuevo término es tal que la suma de todos los pesos de $Rw(c_k)$ sigue siendo 1. Posteriormente si el autor lo desea puede modificar los pesos de la regla haciendo uso de la acción evolutiva “CambiarPesosRw”.

Precondición	Existe una regla de peso asociada al concepto c_k. Esto es, $Rw(c_k) \in Rw(EC_M)$.
	El ítem i_j existe en la estructura conceptual de memorización. Esto es, $i_j \in I(EC_M)$.
	El ítem i_j está asociado funcionalmente al concepto c_k en la EC_M. Esto es, $<c_k, r_f, i_j> \in A_f(EC_M)$, donde $A_f(EC_M)$ representa el conjunto de asociaciones funcionales existentes en EC_M.
	No existe en la regla $Rw(c_k)$ ningún término asociado al ítem i_j. Es decir, $K(c_k.i_j)$ no es un término de la sumatoria expresada en $Rw(c_k)$.
Efecto	La regla de peso $Rw(c_k)$ es trasformada en $Rw'(c_k)$. $Rw'(c_k)$ contiene los mismos términos que $Rw(c_k)$ más el nuevo término $w_j \times K(c_k.i_j)$. La posición del nuevo término $w_j \times K(c_k.i_j)$ en la regla de peso depende del identificador del ítem, esto es $i_{j,.}$ Ya que, para facilitar posteriores búsquedas, los términos de la regla se encuentran ordenados ascendentemente por identificador. Así, elegida la siguiente posición para el nuevo término $w_j \times K(c_k.i_j)$: $Rw'(c_k) : K(c_k) = ... + w_{j-1} \times K(c_k.i_{j-1}) + \mathbf{w_j \times K(c_k.i_j)} + w_{j+1} \times K(c_k.i_{j+1}) + ...$ se satisface que: $i_{j-1} < i_j < i_{j+1}$.
	Los pesos de la regla $Rw'(c_k)$ se recalculan como sigue: - Si la variable DistribuciónPesosRw(c_k) es igual a “ninguna”: El término añadido es el único término de la regla de peso. Por lo tanto, el peso del nuevo término w_j es 1 y a la variable DistribuciónPesosRw(c_k) se le asigna el valor “sistema”. Esto es: $Rw'(c_k) : K(c_k) = 1 \times K(c_k.i_j)$. DistribuciónPesosRw($c_k$) = “sistema”. - Si la variable DistribuciónPesosRw(c_k) es igual a “autor”: El peso del nuevo término se iguala a cero para dejar el resto de pesos tal como los definió el autor, por esto la variable DistribuciónPesosRw(c_k) no se modifica. Esto es: $Rw'(c_k) : K(c_k) = ... + w_{j-1} \times K(c_k.i_{j-1}) + 0 \times K(c_k.i_j) + w_{j+1} \times K(c_k.i_{j+1}) + ...$ Se sugiere al autor que utilice la acción “CambiarPesosRw” para actualizar la distribución de pesos con el nuevo término. - Si la variable DistribuciónPesosRw(c_k) es igual a “sistema”: Todos los pesos se fijan como 1/n, donde *n* es el número total de términos en $Rw(c_k)$, incluyendo el término añadido. La variable DistribuciónPesosRw(c_k) no se modifica porque la distribución de pesos sigue siendo la definida por el sistema. $Rw'(c_k) : K(c_k) = \frac{1}{n} \times \sum_{s=1}^{n} K(c_k.i_s)$.

Salida	No es necesaria ninguna salida ya que esta acción es ejecutada únicamente por el sistema cuando se asocia un nuevo ítem i_j a un concepto c_k existente en la EC_M. Por lo tanto se garantiza la satisfacción de los precondiciones.
Ejecutada_por	Sistema.

5.4 Eliminar un término de una regla de peso

Tabla 6: Acciones evolutivas para modificar las reglas de peso –EliminarTérminoRw–

ACe[4] [Rw]	**EliminarTérminoRw(c_k, i_j)**
Argumentos	**c_k** es el nombre del concepto en cuya regla de peso se desea eliminar un término. **i_j** es el identificador del ítem cuyo término se desea eliminar en $Rw(c_k)$.
Definición	Esta acción evolutiva elimina el término $w_j \times K(c_k.i_j)$ de la regla de peso $Rw(c_k)$. Para lograr que la suma de los pesos de la regla modificada siga siendo igual a 1, el sistema divide el peso w_j del término eliminado entre el número de términos restantes en la regla. Y suma el valor obtenido a cada término de la regla. De este modo, se distribuye de modo uniforme el peso del ítem eliminado entre el resto de ítems.
Precondición	Existe una regla de peso asociada al concepto c_k. Esto es, $Rw(c_k) \in Rw(EC_M)$.
	Existe un término $w_j \times K(c_k.i_j)$ asociado al ítem i_j en la regla de peso $Rw(c_k)$.
Efecto	Independientemente del valor de la variable DistribuciónPesosRw(c_k). - Si la regla de peso $Rw(c_k)$ tiene únicamente el término $w_j \times K(c_k.i_j)$, esto es, $Rw(c_k):K(c_k) = 1 \times K(c_k.i_j)$: La nueva regla queda vacía, por lo que es necesario asignar el valor "ninguna" a la variable DistribuciónPesosRw(c_k). Esto es, $Rw'(c_k): K(c_k) = 0$. Y DistribuciónPesosRw(c_k) = "ninguna". - En otro caso, dada la regla: $Rw(c_k)$: $K(c_k) = \sum_{s=1}^{n} w_s \times K(c_k.i_s)$, la regla de peso resultante después de eliminar el término $w_j \times K(c_k.i_j)$ queda como sigue: $Rw'(c_k): K(c_k) = \sum_{s=1}^{j-1} w_s' \times K(c_k.i_s) + \sum_{s=j+1}^{n} w_s' \times K(c_k.i_s)$ donde $w_s' = w_s + \frac{w_j}{n-1}$.

	Adviértase que si la distribución de pesos es uniforme antes del cambio, después lo sigue siendo, ya que: $w_s' = \frac{1}{n} + \frac{1/n}{n-1} = \frac{1}{n-1}$.
Salida	No es necesaria ninguna salida de control, porque esta acción es ejecutada únicamente por el sistema cuando se elimina la asociación funcional existente en EC_M entre el ítem i_j y el concepto c_k. Por lo tanto, se garantiza la satisfacción de todas sus precondiciones.
Ejecutada_por	Sistema.

5.5 Cambiar los pesos en una regla de peso

Tabla 7: Acciones evolutivas para modificar las reglas de peso –CambiarPesosRw–

ACe[5] [Rw]	**CambiarPesosRw(c_k, <w_1', w_2', ..., w_n'>): boolean**
Argumentos	**c_k** es el nombre del concepto cuya regla de peso se desea modificar. **<w_1', w_2', ..., w_n'>** es la nueva distribución de pesos para la regla de peso Rw(c_k). Está formada por *n* pesos.
Definición	Esta acción evolutiva modifica los pesos asociados a los términos de la regla de peso Rw(c_k). La nueva distribución es asignada por el autor, por lo tanto, después del cambio, el valor de la variable DistribuciónPesosRw(c_k) debe ser igual a "autor".
Precondición	Existe una regla de peso asociada al concepto c_k. Esto es, Rw(c_k) ∈ Rw(EC_M).
	El concepto c_k tiene *n* ítems asociados funcionalmente. O lo que es lo mismo, existen *n* términos en la regla Rw(c_k). Siendo n el número de pesos incluido en la distribución pasada como argumento <w_1', w_2', ..., w_n'>.
	Todos los pesos incluidos en la distribución <w_1', w_2', ..., w_n'> son valores porcentuales. Es decir, $\forall j_1^n : 0 \le w_j' \le 1$.
	La sumatoria de la distribución de pesos <w_1', w_2', ..., w_n'> es igual a 1. Esto es, $\sum_{j=1}^{n} w_j' = 1$
Efecto	La regla de peso Rw(c_k) es trasformada en Rw'(c_k) donde cada peso w_j es sustituido por el nuevo peso w_j' especificado en la distribución que se pasa como argumento. Esto es: La regla $Rw(c_k): K(c_k) = \sum_{j=1}^{n} w_j \times K(c_k.i_j)$,

	se transforma en $Rw'(c_k): K(c_k) = \sum_{j=1}^{n} w_j' \times K(c_k.i_j)$. Además, el valor de la variable DistribuciónPesosRw(c_k) se iguala a "autor", indicando que los pesos de la regla Rw(c_k) han sido definidos por el autor, de acuerdo a algún criterio personal.
Salida	Si alguna de las precondiciones no es satisfecha el resultado de la acción evolutiva es *false*, en otro caso, el efecto es ejecutado y se devuelve *true*.
Ejecutada_por	Autor.

5.6 Reasignar pesos uniformes en una regla de peso

Tabla 8: Acciones evolutivas para modificar las reglas de peso –InicializarPesosRw–

ACe[6] [Rw]	**InicializarPesosRw (c_k)**
Argumentos	**c_k** es el nombre del concepto en cuya regla de peso se va a aplicar la distribución de pesos uniforme.
Definición	Esta acción evolutiva establece los pesos por defecto en la regla de peso para el concepto c_k. Por lo tanto, en la regla modificada, Rw(c_k), se le asigna el mismo peso a todos los ítems i_j ligados a c_k.
Precondición	Existe una regla de peso asociada al concepto c_k. Esto es, Rw(c_k) $\in$ Rw(EC_M).
	La distribución de pesos no es uniforme antes del cambio. Para ello, consultamos la variable DistribuciónPesosRw(c_k) que debe ser igual a "autor".
Efecto	La regla de peso Rw(c_k) se trasforma en la regla Rw'(c_k) donde todos los pesos w_j asociados a los términos de Rw(c_k) son sustituidos en la regla Rw'(c_k) por el valor uniforme 1/n. Así, la nueva regla queda como sigue: $$Rw'(c_k): K(c_k) = \sum_{j=1}^{n} \frac{1}{n} \times K(c_k.i_j)$$ El valor de la variable DistribuciónPesosRw(c_k) se iguala a "sistema", indicando que los pesos de la regla Rw(c_k) han sido definidos por el propio sistema, de acuerdo a la definición de pesos por defecto.
Salida	Si se cumplen las precondiciones exigidas, el sistema devuelve *true* indicando que la distribución de pesos ha sido modificada correctamente. En otro caso se devuelve *false*.
Ejecutada_por	Autor.

5.7 Cambiar el nombre de un concepto en una regla de peso

Tabla 9: Acciones evolutivas para modificar las reglas de peso –CambiarNombreRw–

ACe[7] [Rw]	**CambiarNombreRw (c_k , c_k')**
Argumentos	**c_k** es el nombre del concepto cuyo nombre se desea modificar en la regla de peso correspondiente. **c_k'** es el nuevo nombre para c_k.
Definición	Puesto que el nombre de un concepto es su identificador, si cambia el nombre de un concepto en la EC_M, este cambio debe ser reflejado en su regla de peso. Para ello se utiliza esta acción evolutiva, que sustituye el nombre de un concepto c_k por su nuevo nombre c_k' en la regla de peso que calcula el grado de conocimiento del usuario acerca del citado concepto.
Precondición	Existe una regla de peso asociada al concepto c_k. Esto es, $Rw(c_k) \in Rw(EC_M)$.
	No existe una regla de peso asociada a un concepto con el nuevo nombre c_k'. Esto es, $Rw(c_k') \notin Rw(EC_M)$.
Efecto	El nuevo nombre c_k' sustituye al antiguo nombre c_k en todas sus apariciones en la regla de peso $Rw(c_k)$. Esto afecta tanto a la cabeza de la regla como a los términos del cuerpo. Así, dada la regla: $Rw(c_k): K(c_k) = \sum_{s=1}^{n} w_s \times K(c_k.i_s)$ tras el cambio queda como sigue: $Rw(c_k'): K(c_k') = \sum_{s=1}^{n} w_s \times K(c_k'.i_s)$
Salida	No es necesaria ninguna salida de control, ya que esta acción evolutiva es únicamente ejecutada por el sistema, después de que el nombre de un concepto c_k haya sido modificado por c_k' en la estructura conceptual de memorización. Por lo tanto, se garantiza que se satisfacen las precondiciones exigidas.
Ejecutada_por	Sistema.

5.8 Conjunto de acciones evolutivas

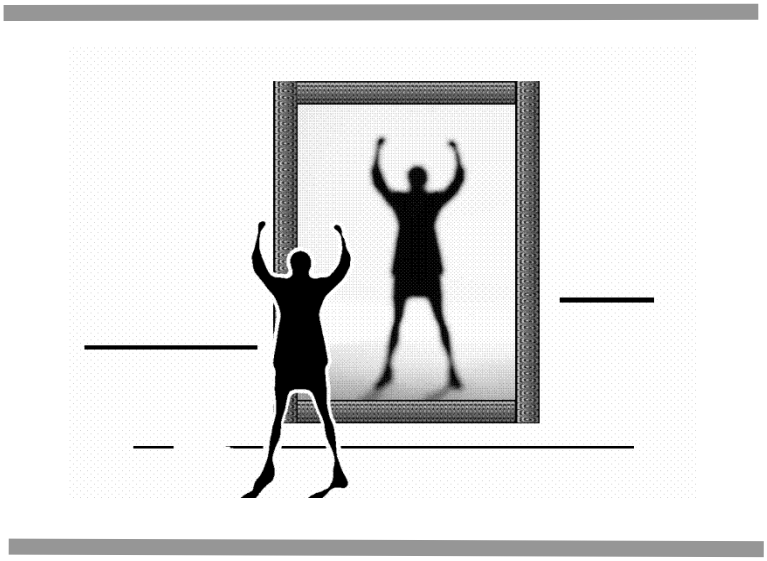
A continuación, en la tabla 10, se resumen todas las acciones evolutivas aplicables sobre las reglas de peso. Para cada acción se describe brevemente su utilidad y las implicaciones que conlleva. También se indica quién tiene la capacidad de ejecutarla, si el sistema o el autor.

Tabla 10: Acciones evolutivas para modificar las reglas de peso

ACe[Rw]	Utilidad/Implicaciones	
CrearRw	Se utiliza para crear la regla de peso de un nuevo concepto. Esto implica que una nueva regla $Rw(c_k)$, inicialmente vacía, sea añadida al conjunto de reglas de la estructura de memorización.	Sistema
EliminarRw	Se utiliza para eliminar la regla de peso de un concepto que deja de existir en la estructura de memorización. Esto implica que dicha regla, $Rw(c_k)$, deje de formar parte del conjunto de reglas $Rw(EC_M)$.	
AñadirTérminoRw	Se utiliza para añadir un nuevo término $K(c_k.i_j)$ a una regla de peso $Rw(c_k)$. Este término corresponde a un ítem i_j asociado recientemente al concepto c_k. Esto implica asignar un peso w_j al nuevo término. Si la distribución de pesos en $Rw(c_k)$ es uniforme, todos los pesos son reajustados para que lo siga siendo. Si la distribución es de autor, el peso w_j debe ser 0 para que no afecte al resto de pesos.	
EliminarTérminoRw	Se utiliza para eliminar un término $K(c_k.i_j)$ de una regla de peso $Rw(c_k)$. Este término corresponde a un ítem i_j que ha dejado de estar asociado al concepto c_k. Esto implica que el peso w_j asociado al término eliminado sea distribuido equitativamente entre el resto de términos. De este modo, si la distribución de pesos era uniforme, tras el cambio lo sigue siendo.	
CambiarNombreRw	Se utiliza para actualizar en la regla de peso de un concepto el cambio de nombre sufrido por éste. Esto implica que el nuevo nombre, c_k', sustituya al antiguo, c_k, en todas sus apariciones en la regla de peso $Rw(c_k)$, que tras el cambio es renombrada como $Rw(c_k')$.	
CambiarPesosRw	Se utiliza para modificar la distribución de pesos existente en una regla $Rw(c_k)$. La nueva distribución debe cumplir las restricciones exigidas en la ecuación 2. Esto implica que los pesos $w_1, w_2, ..., w_n$ existentes en la regla $Rw(c_k)$ sean sustituidos uno a uno por los nuevos pesos especificados en la acción evolutiva $w_1', w_2', ..., w_n'$.	Autor
InicializarPesosRw	Se utiliza para asignar pesos por defecto a los términos de una regla de peso que actualmente tiene una distribución de pesos establecida, de acuerdo a algún criterio, por el autor. Esto implica que la nueva distribución de pesos en la regla $Rw(c_k)$ asigne un peso idéntico, 1/n, a cada uno de sus *n* términos.	

CAPÍTULO 12

Modelo de Usuario



Resumen

En este capítulo se describe la información que el sistema maneja para realizar la adaptación al usuario. Se especifican y detallan los atributos almacenados en el modelo de usuario. Indicando para cada uno de ellos su significado, representación y utilidad. Se define formalmente el esquema y la instancia del modelo de usuario. Se establece la interacción del usuario con la instancia del modelo, y se propone una posible interfaz para su gestión. Del mismo modo se establecen las acciones evolutivas que modifican el esquema del modelo de usuario como consecuencia de los cambios acontecidos en el dominio de conocimiento del sistema hipermedia.

Tabla de contenidos

Modelo de Usuario

1. INTRODUCCIÓN

En cualquier proceso de adaptación es esencial conocer determinadas características sobre el objeto de dicha adaptación. Estas características serán distintas dependiendo del tipo de sistema y a qué o quién se adapta. Pero, aún sin concretar qué información es necesaria, parece indiscutible que en un sistema hipermedia adaptativo es imprescindible conocer algo (mejor si es mucho) sobre el individuo que lo utiliza.

> **Def 12.1 [Modelo de usuario]** El modelo de usuario, al que de forma abreviada notamos MU, es precisamente, la *representación interna que el sistema mantiene del usuario* para posteriormente ser capaz de adaptar su estructura y comportamiento a las características y necesidades del mismo.

Según Brusilovsky en [Brusilovsky, 96]:

> "Todo sistema hipermedia adaptativo debe satisfacer tres criterios: ser un sistema hipermedia o hipertexto, tener un modelo de usuario y ser capaz de adaptar el hipermedia usando este modelo".

La definición anterior confirma la importancia del modelo de usuario en el proceso de adaptación, siendo fundamental que el sistema sea capaz de mantenerlo lo más completo y exacto posible. Preocupándose de actualizarlo a medida que cambian las características y necesidades del usuario.

Por tanto, el modelo de usuario es almacenado, mantenido y consultado por el sistema, con el fin de adaptarse a cada usuario, proporcionándole la navegación y presentación de la información que necesita. Para ello, el modelo de usuario contiene, en cada momento, el estado actual del usuario. Dicho estado representa no sólo el conocimiento adquirido por éste, sino también sus intereses, preferencias y experiencias previas.

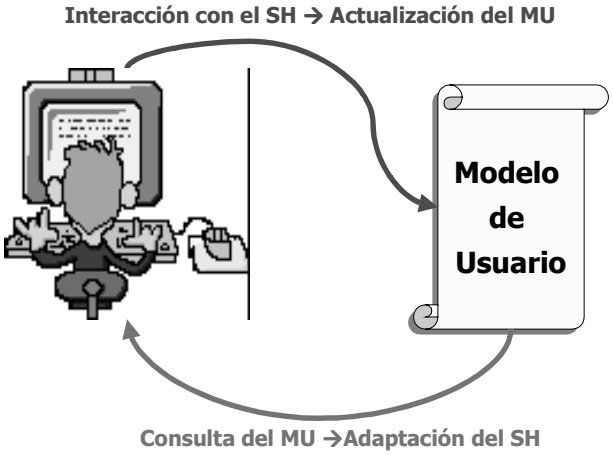


Figura 1: Construcción y uso del MU

En la figura 1, se muestra cómo la interacción del usuario con el sistema hipermedia alimenta su modelo con nueva información, o actualiza alguna información ya existente.

Y como la información contenida en el modelo acerca del usuario es consultada por el sistema para adaptar y mejorar la interacción de éste. Se trata pues, de un proceso cíclico que permite personalizar la navegación del usuario más y mejor a medida que éste trabaja con el sistema.

2. GESTIÓN DEL MODELO DE USUARIO: INICIALIZACIÓN Y ACTUALIZACIÓN

Dentro de la información contemplada en el modelo de usuario, hay atributos cuyo valor sólo puede conocer el sistema a través de una **petición explícita** al usuario, mientras que otros, pueden ser **automáticamente deducidos** mientras que el usuario recorre, accede y lee la información ofrecida por el sistema hipermedia.

Para inicializar de forma fiable el modelo de usuario, la mayoría de los atributos se consultan al usuario la primera vez que éste se registra en el sistema. A partir de ahí algunos de estos atributos se modifican únicamente cuando el usuario lo requiere, mientras que otros, son actualizados por el sistema de acuerdo con la interacción del usuario y con determinados criterios previamente establecidos por el autor. Por ejemplo, el conocimiento adquirido por el usuario durante la lectura de los ítems de información se obtiene aplicando las reglas de actualización y conocimiento definidas por el autor (capítulos 8 y 9).

Como hemos mencionado, consideramos que una inicialización explícita es más fiable, ya que permite evitar futuras adaptaciones inadecuadas, basadas en una información inicial defectuosa. No obstante, el sistema debe ser capaz de realizar una inicialización por defecto de la mayoría de los atributos del modelo de usuario. Esta capacidad consiente el uso del sistema sin tener que realizar ninguna inicialización explícita, librando al usuario de esta tarea si así lo desea.

En nuestro sistema, el modelo de usuario es inicializado mediante una petición explícita a través de una especie de formulario, donde el usuario debe rellenar los datos requeridos. De forma que, aquellos datos que no proporciona, son sometidos a una inicialización por defecto. Después, determinadas características se actualizan automáticamente mientras que otras sólo cuando el usuario lo solicita.

En el caso de las características actualizadas bajo demanda del usuario, hay algunas para las que es imprescindible consultarle a éste el nuevo valor, mientras que otras admiten las dos posibilidades: que el nuevo valor sea proporcionado por el usuario o inferido por el sistema. Además, para que el usuario tenga cierto *control sobre la adaptación*, en cualquier momento puede modificar el valor de una característica actualizada automáticamente si observa que la deducción realizada no es correcta.

Ese control explícito del modelo por parte del usuario hace que éste incremente su confianza en el sistema, ya que tiene en sus manos la capacidad de detectar y subsanar inferencias erróneas, lo cual repercutirá en una adaptación más acertada. Sin embargo, puede ser un arma de doble filo.

Por ejemplo, en aplicaciones educacionales los alumnos podrían hacer trampas, quizá modificando a la alza la inferencias realizadas por el sistema sobre su estado de conocimiento, con el objeto de satisfacer sin esfuerzo ciertos requisitos de conocimiento y llegar a una meta impuesta por el profesor.

Por lo tanto, el sistema debe dar la posibilidad de que sea el autor quién determine si debe estar habilitada o no esta capacidad, en función del tipo de aplicación y el público al que está dirigida.

3. ELEMENTOS DEL MODELO DE USUARIO

En la sección anterior se ha expuesto que la intervención del usuario en la gestión de su propio modelo es tanto directa (actualización explícita) como indirecta (actualización automática a partir de su interacción con el sistema). Por lo tanto, podemos calificar nuestro sistema hipermedia como *adaptable y adaptativo* [Medina, 02e]

En la sección actual, se detallan, uno a uno, el conjunto de atributos incluidos en el modelo de usuario para representar el estado actual de éste. Para cada atributo o característica se indica: nombre, significado, utilidad, representación, valor por defecto y actualización. Para su descripción, los atributos se presentan agrupados en cinco categorías: datos personales, conocimiento, experiencia, preferencias e intereses. Los atributos contemplados en cada categoría se enumeran en la tabla 1. Cada categoría se explica en una sección diferente.

Tabla 1: Atributos del modelo de usuario

Categoría	Atributos
Datos personales	Clave de acceso, nombre, apellidos, sexo, edad y ocupación.
Conocimiento	Número de visitas y grado de conocimiento.
Experiencia	Experiencia en la materia y experiencia de navegación.
Preferencias	Preferencias sobre ítems, preferencia por rutas cortas y preferencias para la estructura de los resúmenes.
Intereses	Subdominio de conocimiento, ítems y conceptos interesantes, y meta de conocimiento.

Cada uno de estos atributos ha sido incluido en el modelo de usuario porque necesitamos esa información para realizar un determinado tipo de adaptación. Junto a la descripción de cada atributo se explica brevemente su utilidad principal, que es ampliada y desarrollada en el capítulo dedicado al modo de navegación donde se lleva a cabo la adaptación que lo utiliza (modulo IV, Elementos Funcionales del Sistema de Aprendizaje).

Revisando la literatura científica podemos encontrar que muchos de estos atributos han sido incluidos, de un modo más o menos similar, en el modelo de usuario de otros sistemas adaptativos. En ADAPTS [Brusilovsky, 99], por ejemplo, el modelo de usuario contiene la experiencia, el conocimiento y las preferencias del usuario en relación a tareas técnicas y componentes hardware. En AHM [DaSilva, 97] el modelo de usuario recoge el conocimiento sobre los conceptos del modelo de dominio. En el sistema adaptativo propuesto por [Hijikata, 01] para asistir a los proveedores de información se almacena la historia de navegación del usuario, guardando no sólo los nodos visitados sino también el orden de las visitas.

AHAM [DeBra, 98] es un modelo de adaptación general, por lo tanto, también, define un modelo de usuario muy completo que contiene información acerca del conocimiento del usuario sobre un concepto, si el usuario ha leído algo sobre éste o si está preparado

para entenderlo, las preferencias del usuario sobre el esquema de coloreo de enlaces, sus metas, experiencia en la materia y experiencia en la navegación.

En ningún trabajo realizado hemos encontrado atributos que hagan referencia a la longitud de las rutas de navegación, al esquema de composición de los resúmenes, o al subdominio de conocimiento que interesa al usuario. Además, aunque el conocimiento y el número de visitas es un atributo ampliamente utilizado, pocos autores mantienen esta información a doble nivel, conceptos e ítems, como es nuestro caso.

3.1 Datos personales

Los datos personales referentes al usuario que se recogen en el modelo son: nombre, apellidos, sexo, edad y ocupación. Junto a éstos, también se almacena una clave de acceso que el usuario debe utilizar para registrarse en el sistema y que lo identifica unívocamente del resto.

La representación de estos datos es sencilla. Nombre, apellidos y ocupación son trozos de texto, es decir cadenas de caracteres. La edad debe ser un número entero entre un valor mínimo, por ejemplo 5, y un valor máximo, por ejemplo 100. El atributo sexo sólo puede tomar dos valores distintos, 'hombre' o 'mujer'. Por su parte la clave de acceso es una secuencia alfanumérica que deberá cumplir unas normas estándar de seguridad, como por ejemplo tener un número mínimo de caracteres o combinar letras y dígitos.

El sistema utiliza, en su mayoría, los datos personales para dirigirse de una forma personalizada al usuario, es decir, los mensajes y avisos del sistema pueden encabezarse con el nombre del usuario haciendo más familiar el proceso de comunicación. Incluso puede cambiarse el talante de dichos mensajes en función de la edad del usuario o su ocupación, lo cual descubre un uso del sistema aún más personalizado y agradable.

Además de la utilidad expresada más arriba, determinados datos personales pueden utilizarse para deshabilitar la capacidad del usuario de modificar las inferencias hechas por el sistema. Por ejemplo, la edad puede ser un buen indicador para restringir la capacidad de control sobre el sistema.

El autor podría establecer que los usuarios cuya edad sea inferior a 16 años no pueden modificar las actualizaciones realizadas de forma automática en su modelo.

Cabe hacer notar que los datos personales son la única sección del modelo de usuario que no tiene inicialización por defecto. Este tipo de datos se actualiza únicamente bajo la solicitud explícita del usuario con el nuevo valor proporcionado por éste o con el valor recogido de alguna fuente de información, por ejemplo de una base de datos. Esto se debe a que, aunque se pudiese recurrir a complicadas teorías psicológicas para "adivinar", a partir de la interacción en el sistema, el rango de edad del usuario o incluso su sexo, existen datos como el nombre y apellidos sobre los que es imposible realizar conjeturas.

3.2 Conocimiento

Dentro de esta sección incluimos el estado de conocimiento del usuario sobre los ítems y los conceptos del sistema hipermedia. Concretamente, para cada ítem y concepto

presente en la estructura conceptual de memorización EC_M existe una entrada en el modelo de usuario. En dicha entrada se almacena no sólo el **grado de conocimiento** que el usuario posee sobre ese ítem o concepto, sino también el número de **veces que lo ha visitado**. Nótese, que se incluyen en el modelo de usuario todos los conceptos e ítems del Sistema de Memorización, con independencia de las presentaciones que después navegue el usuario.

3.2.1 Número de visitas

El número de visitas realizadas a un ítem o concepto, se representa con un número entero mayor o igual que cero. Un número de visitas igual a cero para un ítem i_j, visitas(i_j) = 0, implica que el usuario nunca ha accedido al contenido de dicho ítem. Mientras que para un concepto c_k, visitas(c_k) = 0, significa que el usuario no ha solicitado nunca un resumen de éste durante la navegación por conceptos (capítulo 16).

El número de visitas realizadas a los ítems de información es consultado por el sistema durante la navegación por relación conceptual (capítulo 17), para adaptar la navegación del usuario en función de los ítems que éste ha visitado previamente. Además, como veremos más adelante, esta información puede servir como un indicador parcial de la experiencia del usuario en la navegación de sistemas hipermedia.

Por supuesto, la inicialización de este atributo debe ser cero para todos los ítems y conceptos de la EC_M. Este valor inicial refleja el hecho de que el usuario no ha accedido todavía al sistema, por lo que es realmente improbable que haya tenido acceso a los trozos de información que en éste se ofrecen.

La actualización del número de visitas realizadas sobre un ítem i_j se realiza de forma automática incrementando en uno el número de visitas sobre éste cada vez que el usuario inspecciona su contenido. Esto es, visitas(i_j) = visitas(i_j) + 1. Lo mismo ocurre con el número de visitas de un concepto, cada vez que el usuario accede a la información del resumen compuesto para éste.

3.2.2 Grado de conocimiento

El número de visitas efectuadas a un ítem está relacionado con el grado de conocimiento que el usuario posee sobre éste. Sin embargo, no se trata de una relación determinante. Recuerde, que las reglas de actualización (capítulo 9) permiten que sin haber visitado nunca un ítem sea posible obtener conocimiento sobre él, a través del estudio de otros ítems relacionados. Además, de acuerdo a las reglas de conocimiento (capítulo 8), visitar un ítem no supone siempre adquirir conocimiento sobre él. Esto sólo sucede si se posee el conocimiento previo exigido en sus restricciones de accesibilidad. Incluso en dicho caso, aunque sea lo más frecuente, una sola visita al ítem puede no bastar para lograr conocimiento "total" sobre él.

Por este motivo, además del número de visitas realizadas sobre un ítem i_j es necesario guardar el grado de conocimiento que el usuario ha logrado sobre éste como consecuencia de su lectura y, quizás, la de otros ítems relacionados. Este grado (definiciones 8.3 y 8.4) es representado mediante la función de conocimiento K(i_j) (definición 8.5) que admite cinco valores distintos: 0 ("nulo"), 1 ("bajo"), 2 ("medio"), 3 ("alto") y 4 ("total").

Este atributo es uno de los más importantes, ya que permite evitar numerosos problemas de falta de comprensión al ser utilizado para adaptar la navegación del usuario en función del estado de conocimiento que posee. Este proceso de adaptación se explica con más detalle en el capítulo dedicado a la navegación por conocimiento (capítulo 18). Además, en la siguiente sección, se expone cómo el grado de conocimiento del usuario sobre los conceptos puede ser un indicador muy útil acerca de la experiencia del usuario en la materia de dicho sistema hipermedia.

Inicialmente el grado de conocimiento es 0, o sea "nulo", para todos los ítems incluidos en la estructura conceptual de memorización. La inicialización por defecto (véase en la ecuación 1), supone que el usuario no ha estudiado previamente ninguno de los trozos de información que constituyen el sistema. Puesto que el grado de conocimiento sobre los conceptos se obtiene aplicando las reglas de peso (capítulo 11), sea cual sea la distribución de pesos en una regla $Rw(c_k)$, el conocimiento inicialmente obtenido para el concepto c_k es también "nulo".

$$\forall i_j \in I(EC_M) \rightarrow K(i_j) = 0. \qquad \forall c_k \in C(EC_M) \rightarrow K(c_k) = 0. \qquad (1)$$

La actualización de este atributo se realiza automáticamente atendiendo a la navegación del usuario, al conjunto de reglas de actualización (Ru) establecidas en la estructura conceptual de aprendizaje, EC_A^i, que actualmente se navega, y al conjunto de reglas de peso definidas sobre la EC_M. Por lo tanto depende de la navegación del usuario y de las reglas de adquisición de conocimiento previamente creadas por el autor.

Concretamente cada vez que el usuario visita un ítem i_j, y siempre que las reglas de conocimiento así lo deciden, el sistema aplica la regla de actualización $Ru(i_j)$ para obtener el nuevo grado de conocimiento sobre ése, $K'(i_j)$, y otros ítems i_k actualizados en el cuerpo de la regla, $K'(i_k)$.

Cuando cambia el grado de conocimiento sobre un ítem, puede ocurrir que el grado de conocimiento sobre alguno de los conceptos a los que éste se asocia también cambie. Así, el grado de conocimiento sobre un concepto c_k debe ser reconsiderado cada vez que varía el grado de conocimiento del usuario sobre algún ítem i_j asociado funcionalmente a él. Para ello, el sistema vuelve a aplicar su regla de peso, $Rw(c_k)$, con el nuevo valor obtenido para el término $K(c_k.i_j)$, esto es $K'(i_j)$.

Al tratarse de un atributo calculado, el grado de conocimiento sobre un concepto, $K(c_k)$, se vuelve a determinar cada vez que cambia el conocimiento acerca de los ítems implicados en el cálculo. Asimismo, también debe ser computado de nuevo, si cambia la forma de realizar dicho cálculo, esto es, la distribución de pesos de la regla $Rw(c_k)$.

Por lo tanto, el uso de las acciones evolutivas: CambiarPesosRw, InicializarPesosRw, EliminarTérminioRw, o AñadirTérminioRw sólo cuando la distribución de pesos es uniforme, desencadenan un seudoproceso de propagación interna, que no afecta a la estructura del modelo de usuario, sino a las distintas instanciaciones de éste. Es decir, en estos casos, la distribución de pesos cambia y el sistema aplica la regla de peso modificada, $Rw'(c_k)$, para obtener y actualizar el nuevo grado de conocimiento acerca del concepto c_k en todos los modelos usuarios existentes.

3.3 Experiencia

El modelo de usuario contempla dos tipos distintos de experiencia: **experiencia en la materia** y **experiencia de navegación**. En ambos tipos, el grado de experiencia se representa mediante una de las etiquetas semánticas: "nulo", "bajo", "medio", "alto", "total", utilizadas también para expresar los grados de conocimiento. Tanto la experiencia en la materia, como la experiencia de navegación, son empleadas, junto con otros atributos del modelo de usuario, para determinar qué estructura de navegación es la más adecuada para el usuario. Los detalles de cómo toma el sistema esta decisión son descritos en el capítulo 14, Elección de la EC_A.

3.3.1 Experiencia en la materia

La experiencia en la materia refleja el conocimiento general que el usuario posee sobre el dominio conceptual del sistema hipermedia, esto es, los conceptos y asociaciones conceptuales incluidas en la EC_M.

> **Def 12.2 [Experiencia en la materia]** La experiencia en la materia trata de identificar con qué grado conoce o cómo está de familiarizado el usuario con el conjunto de conceptos sobre los que versa el material ofrecido en el sistema hipermedia, es decir, cuál es su experiencia en el tema desarrollado en dicho sistema.

Inicialmente, por defecto, la experiencia en la materia del usuario es "nula". Posteriormente, este dato sólo es modificado cuando el usuario lo solicita explícitamente, bien proporcionando un nuevo valor o pidiendo una actualización automática.

En el último caso, para averiguar esta información, el sistema calcula el grado de conocimiento medio que el usuario posee sobre los conceptos del sistema. Es decir, si la experiencia en la materia del usuario debe reflejar su conocimiento global acerca de los conceptos del sistema, un posible indicador puede ser obtenido a partir de su conocimiento individual sobre cada concepto. Para ello, el sistema utiliza el valor del atributo grado de conocimiento, $K(c_k)$, asociado en su modelo de usuario a cada concepto c_k existente en la EC_M. La formula utilizada se muestra en la ecuación 2.1.

$$\exp eriencia - materia = \frac{\sum_{k=1}^{m} K(c_k)}{m} \text{ donde } c_1, c_2, ..., c_m \in C(EC_M) \qquad (2.1)$$

El resultado obtenido es un número real, mayor o igual que 0 y menor o igual que 4, que debe ser aproximado al valor entero más próximo, para hacerle corresponder una de las cinco etiquetas semánticas (0 –"nulo", 1 – "bajo", 2 – "medio", 3 – "alto", 4 – "total").

Para ganar precisión, el valor utilizado en el cálculo del atributo experiencia-materia puede ser directamente el resultado de ejecutar las reglas de peso de los conceptos, $Rw(c_k)$, sin aplicarles el redondeo que convierte el resultado de la regla en un número entero válido para $K(c_k)$. En este caso, la fórmula aplicada se muestra en la ecuación 2.2.

$$\exp eriencia - materia = \frac{\sum_{k=1}^{m} Rw(c_k)}{m} \text{ donde } c_1, c_2, ..., c_m \in C(EC_M) \qquad (2.2)$$

La única aproximación que se realiza ahora corresponde al redondeo final aplicado sobre el valor real obtenido como resultado de la fórmula. Esta alternativa permite ganar exactitud, pero es menos eficiente que la anterior si hay que recalcular la regla de peso de cada concepto en lugar de utilizar directamente el valor del atributo $K(c_k)$. Por este motivo, puede ser interesante conservar el resultado de $Rw(c_k)$ antes de aplicar el redondeo. En cualquier caso, la pérdida de exactitud sufrida en el redondeo de cada $Rw(c_k)$ suele ser compensada en media, de forma que normalmente no se pierde apenas precisión si se calcula la experiencia-materia como se establece en la ecuación 2.1.

3.3.2 Experiencia de navegación

La experiencia de navegación determina lo acostumbrado que está el usuario a seguir enlaces entre nodos hipermedia, moverse a través de páginas web, mantener en paralelo varias secuencias de visita, volver a un punto anterior cuando un camino no le convence, etc.

> **Def 12.3 [Experiencia de navegación]** La experiencia de navegación representa la práctica del usuario en el uso de sistemas hipermedia.

Esta información es importante para saber cuán compleja puede ser la estructura de enlaces proporcionada a un usuario, con la intención de minimizar los problemas de sobrecarga cognitiva y desorientación.

Al igual que en el caso anterior, por defecto, la experiencia de navegación es inicialmente “nula” y únicamente se actualiza bajo demanda del usuario. Puesto que la experiencia de navegación depende de la interacción del usuario con el sistema actual, pero también, y en gran parte, de su experiencia anterior, siempre se prefiere una actualización explícita, ya que la actualización automática está inevitablemente limitada al uso que el usuario hace del sistema actual.

No obstante, también se ofrece una actualización automática para la experiencia de navegación. Esta actualización se realiza teniendo en cuenta el número de ítems visitados por el usuario en el sistema. Permítanme repetir, que se trata de un indicador parcial, ya que sólo informa de la habilidad presentada por el usuario para navegar dentro de este sistema concreto. Cuando, en realidad, la experiencia de navegación debería reflejar la pericia del usuario adquirida a lo largo de los recorridos realizados en todos los sistemas hipermedia utilizados.

La ecuación 3 muestra la fórmula que aplica el sistema para estimar este tipo de experiencia. El procedimiento consiste en dividir el número de ítems visitados por el usuario en el sistema entre el número total de ítems que existen en el mismo. El resultado de la división, es un número real mayor o igual que 0 y menor o igual que 1. El cuál representa el porcentaje de ítems del sistema visitados por el usuario. Dicho valor coincide con 1 únicamente cuando el usuario ha visitado todos los ítems del sistema.

El valor obtenido en el rango [0,1] se multiplica por una constante *C* que traslada el resultado de la fórmula al rango [0, C]. El valor de *C* debe ser, por tanto, menor o igual

que 4, ya que este es el grado de experiencia máximo. Dicho valor es establecido por el autor para indicar el grado de experiencia que quiere hacer corresponder a un usuario que ha visitado todos los ítems de su sistema.

$$\text{exp}\,eriencia - navegación = C \times \frac{\sum_{j=1}^{n} visitado(i_j)}{n} \qquad (3)$$

donde:

- *n* es el número total de ítems existentes en la estructura conceptual de memorización, EC_M.
- $i_1, i_2, ..., i_n$ son dichos ítems. Esto es, $I(EC_M) = \{i_1, i_2, ..., i_n\}$.
- *visitado*(i_j) es una función que devuelve 1 si el número de visitas realizadas al ítem i_j es mayor que cero y 0 en otro caso. Para obtener el resultado de la función, el sistema consulta el atributo visitas(i_j) incluido en el modelo del usuario.

$$\text{visitado}(i) = \begin{cases} 0 & \text{si visitas}(i_j) = 0 \\ 1 & \text{si visitas}(i_j) > 0 \end{cases}$$

- *C* es una constante real positiva menor o igual a 4, $C \in [0, 4]$, que establece el grado de experiencia de navegación que tiene un usuario que ha visitado, al menos una vez, los *n* ítems de la estructura conceptual de memorización.

Por defecto el valor de la constante *C* es igual a 4, con lo que se asigna un grado de experiencia de navegación "total" a un usuario que visite todos los ítems del presente sistema. Sin embargo, el autor puede definir un valor menor para *C*, si considera que visitar todos los ítems de su sistema hipermedia no concede al usuario un grado de experiencia "total" en la navegación web. Cuanto más pequeño sea el sistema hipermedia, menor debería ser el valor de la constante *C*.

Por ejemplo, supongamos que el autor asigna valor 3 a la constante C.

Si un usuario ha visitado 65 de los 100 ítems existentes en dicho sistema hipermedia, el cálculo automático de su experiencia de navegación es el siguiente:

$$\text{experiencia-navegación} = 3 \times 0{,}65 = 1{,}95 \approx 2 \text{ (“medio”)}$$

Adviértase que, de nuevo, el resultado final de la fórmula ha sido redondeado al valor entero más próximo para poder asignarle una etiqueta semántica.

Por supuesto, si el usuario no está conforme con la inferencia realizada no tiene más que establecer explícitamente cuál considera él que es su experiencia de navegación.

3.4 Preferencias

No sólo es importante saber cuál es el estado de conocimiento del usuario o qué experiencia previa posee. También es conveniente conocer sus gustos y preferencias

para adaptarse a éstas, en la medida de lo posible, haciendo más agradable la interacción del usuario con el sistema.

Concretamente son de tres tipos diferentes las preferencias contempladas en el modelo de usuario: **preferencias sobre ítems**, **preferencia por rutas cortas** y **preferencias para la estructura de los resúmenes**.

3.4.1 Preferencias sobre ítems

Las preferencias sobre ítems se refieren a los *gustos del usuario acerca de los ítems* o trozos de información del sistema hipermedia. Concretamente el usuario puede especificar sus preferencias sobre las seis características o atributos que se indican en la tabla 2.

Tabla 2: Atributos de un ítem

Característica	Descripción
idioma	Lengua en la que se encuentra el texto o audio contenido en el ítem.
autor	Información sobre el autor del ítem.
fecha-de-creación	Fecha en la que se elaboró el ítem o su última actualización.
nivel-de-dificultad	Grado de dificultad o sencillez con el que se expresa la información almacenada en el ítem.
medio	Medio utilizado para representar la información: texto, imagen, audio, video, ejecutable, etc.
rol	Papel informativo que desempeña el ítem: introducción, ejemplo, definición, etc.

Para cada una de estas características el sistema proporciona al usuario una lista con los valores que permite. Esa lista agrupa todos los valores diferentes que para dicha característica presenta algún ítem de la estructura conceptual de memorización. Excepto para el atributo fecha-de-creación, donde se consideran periodos de tiempo en lugar de fechas concretas para hacer la lista menos extensa.

Inicialmente y por defecto, para cada característica, la inclinación del usuario por cada uno de sus posibles valores es indiferente. Después, si lo desea, el usuario puede especificar sus preferencias sobre cada característica, marcando los valores que le gustan con un *SI* y los que no le gustan con un *NO*. Además, puede ordenar sus preferencias (ya sean positivas o negativas), dependiendo de cuánto le guste o disguste cada una. El orden es descendente, de forma que el gusto colocado en la posición *i* tiene más importancia que el situado en la posición *i+1*. El usuario puede colocar dos preferencias al mismo nivel, asignándoles la misma posición en el orden, cuando su apego por ambas sea similar.

Además de especificar para cada característica los valores que sí le gustan, los que no le gustan y en qué orden, el usuario puede establecer la importancia que para él tiene cada una de las características de la tabla 2. Para ello, simplemente asigna un peso a cada uno de estos atributos, de modo que la suma de todos ellos sea 1.

Así, si definimos el conjunto de atributos A = {idioma, autor, fecha-de-creación, nivel-de-dificultad, medio, rol}, y notamos característica^i al atributo que ocupa la posición *i*

en dicho conjunto, con el valor de i desde 1 hasta 6. Debe cumplirse que $\sum_{i=1}^{6} peso(característica^i) = 1$.

En caso de que la suma de la distribución de pesos establecida por el usuario para los elementos de *A* no satisfaga la condición anteriormente expresada. El sistema aplica, si el usuario así lo desea, un proceso de normalización automático que se muestra en la ecuación 4.

$$\forall i = 1..6, \quad \text{peso}'(\text{característica}^i) = \frac{peso(característica^i)}{\sum_{j=1}^{6} peso(característica^j)} \qquad (4)$$

Tras el proceso indicado, el peso proporcionado por el usuario para el atributo característica^i, esto es peso(característica^i), es transformado en peso'(característica^i), de tal forma que ahora $\sum_{i=1}^{6} peso'(característica^i) = 1$.

Por defecto, todas las características tienen la misma importancia, esto es, la unidad se reparte entre los seis atributos, dando lugar a un peso de 1/6 para cada uno. Sin embargo, a través del proceso de ponderación, el usuario puede determinar que características de un ítem son para él primordiales (peso alto) y cuales accesorias o prescindibles (peso bajo).

En la figura 2 se muestra un ejemplo, donde el usuario especifica sus preferencias respecto a cinco de las seis características de un ítem: idioma, medio, rol, nivel-de-dificultad y fecha-de-creación.

Idioma		
1	Sí	español
1	Sí	inglés
2	No	ruso
3	Sí	italiano

Medio		
1	Sí	texto
2	Sí	imagen
3	No	audio

Rol		
1	Sí	definición
2	No	opinión
2	No	aclaración
3	Sí	ejemplo
4	No	introducción

Dificultad		
1	Sí	sencillo

Fecha		
1	Sí	2003
2	No	anterior a 1980

Autor		

peso(Idioma) = 0,3	peso(nivel-de-dificultad) = 0,1
peso(Medio) = 0,1	peso(Fecha-de-creacion) = 0,2
peso(Rol) = 0,1	peso(Autor) = 0,2

Figura 2: Preferencias de un usuario

Por ejemplo, respecto a la característica *idioma* declara que le gusta el "español", "inglés" e "italiano" y le disgusta el "ruso". Que el "español" y el "inglés" le gustan por igual y que el "italiano" le gusta menos que el "español" o el "inglés". Que le gusta más

el “español” y el “inglés” de lo que le disgusta el “ruso”. Y que le disgusta el “ruso” más de lo que le “gusta” el italiano.

Además, establece una asignación de pesos para las seis características (0,3+0,1+0,1+0,1+0,2+0,2 = 1) distinta de la asignación por defecto (1/6+1/6+1/6+1/6+1/6+1/6 = 1). De los pesos proporcionados se deduce que el idioma en que se presenta la información del ítem es muy importante para este usuario.

3.4.2 Preferencia por rutas cortas

Como se explica más adelante, en el capítulo 20 (Rutas guiadas), las preferencias del usuario son utilizadas, principalmente, para proporcionar a éste una ruta personalizada a través del espacio de navegación. El objetivo de la ruta es asegurar la consecución por parte del usuario de una determinada meta de conocimiento. Las preferencias son tenidas en cuenta, para conseguir que el recorrido de la ruta incluya ítems cuyas características se ajusten lo mejor posible a los gustos especificados por el usuario.

Pero, para que el sistema pueda elegir la mejor ruta, el usuario debe establecer su gusto respecto a una característica más, que no se refiere a las propiedades de los ítems visitados en la ruta, sino a la ruta en sí. Este nuevo atributo, denominado **ruta-corta**, se refiere a la *longitud de la ruta* y refleja la importancia que el usuario concede a que el número de visitas realizadas en la ruta sea lo más pequeño posible.

El usuario debe establecer un peso entre 0 y 1 para dicho atributo, esto es: $0 \leq$ peso(ruta-corta) ≤ 1. Dicho peso debe estar más próximo a 1 cuanto mayor sea la preferencia del usuario por las rutas cortas. Coincidiendo con 1 si desea obtener la ruta más corta que exista, independientemente de cuáles sean las características de los ítems incluidos en ella.

Obviamente, la elección de un peso alto para el atributo ruta-corta va a ir en decremento de las preferencias establecidas por el usuario acerca de las características de los ítems (idioma, autor, etc.). De forma que, según cuál sea el peso asignado al atributo ruta-corta, quede más o menos peso para repartir entre las seis características incluidas en *A* (tabla 2). Notamos peso(preferencias-sobre-ítems) a la suma de esos seis pesos. Esto es, peso(preferencias-sobre-ítems) = $\sum_{i \in A}$ peso(característica^i).

Inicialmente, cuando el peso de ruta-corta es 0, el peso disponible para **preferencias-sobre-ítems** es 1. Esto es, peso(ruta-corta) = 0 $\Rightarrow$ peso(preferencias-sobre-ítems) = 1. Sin embargo, cuando el peso establecido para el atributo ruta-corta es mayor que 0, el peso total del que se dispone para las preferencias sobre ítems es 1 menos el peso(ruta-corta). De forma que, el peso asignado por el usuario a cada característica incluida en *A*, debe ser reajustado en función del peso total adjudicado a preferencias-sobre-ítems. Este reajuste es realizado automáticamente por el sistema tal y como se propone en la ecuación 5.

$$\text{peso(ruta-corta)} = w \Rightarrow \text{peso(preferencias-sobre-ítems)} = (1-w) \qquad (5)$$
$$\forall_{i \in A} \ \text{peso'(característica}^i\text{)} = \text{peso(característica}^i\text{)} \times \text{peso(preferencias-sobre-ítems)}.$$

Retomando el ejemplo de la figura 2, supongamos que el usuario pondera el atributo ruta-corta con un valor 0,3 (en lugar del valor inicial 0).

En esta nueva situación, el peso total para las preferencias-sobre-ítems es reducido de 1 a 0,7. Por lo que el peso asociado a cada característica es reajustado de forma que se sigue manteniendo la proporción entre los pesos de las distintas características, pero la suma total de los pesos es ahora 0,7.

La distribución de pesos inicial y resultante se muestra en la figura 3.

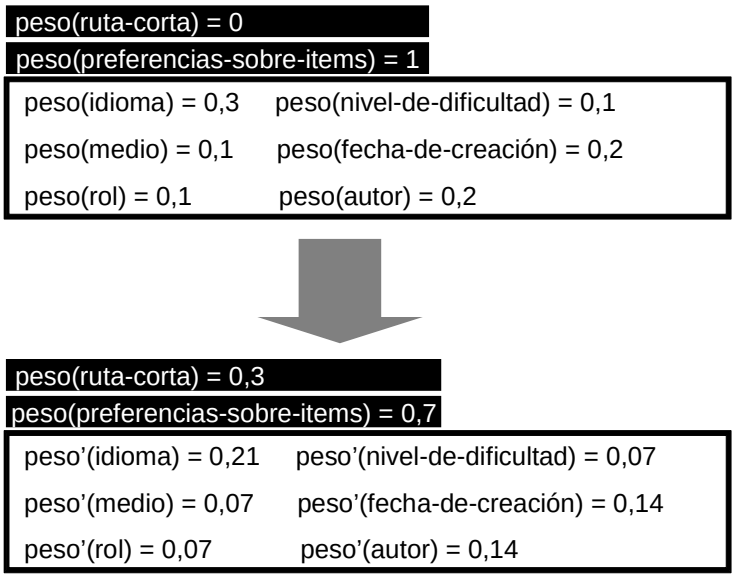


Figura 3: Nueva distribución de pesos para las preferencias sobre un ítem

3.4.3 Preferencias para la estructura de los resúmenes

Tal y como se describe, con más detalle, en el capítulo 16 (Navegación por conceptos), el usuario puede navegar sobre una estructura donde se le muestran únicamente los conceptos y relaciones conceptuales existentes. Esto le permitirá centrarse en los conceptos, despreocupándose en un primer momento de la información concreta asociada a ellos. En tal caso, el usuario visita conceptos, en lugar de ítems. De modo que al seleccionar un concepto es cuando obtiene la información relativa a éste. Dicha información es un *resumen compuesto con los ítems asociados al concepto*.

La **estructura de composición** del resumen es inicialmente definida por el autor. Para indicar la forma en que se organiza la información en los resúmenes, el autor no tiene que fijar para cada concepto el orden en que aparecerán sus ítems. Sino que establece, de forma general, el orden en que deben aparecer los ítems de cualquier concepto, en función del rol con el que se asocian a éste. Una vez establecido un orden estricto entre todos los roles posibles, el autor debe indicar la importancia de cada rol en el resumen, distinguiendo qué roles son obligatorios y cuáles opcionales.

Finalmente, en el resumen de un concepto aparece una sección por cada rol en el orden fijado. Las *secciones opcionales* se encuentran comprimidas y las *secciones obligatorias* visualizan el contenido de todos los ítems asociados con ese rol al concepto.

Para obtener resúmenes personalizados, el usuario debe poder establecer sus preferencias acerca de la estructura de los mismos. Inicialmente, el resumen es

construido con la estructura fijada por el autor, pero después el usuario puede modificar dicha estructura de dos formas diferentes:

a) Explícitamente en el modelo de usuario, el usuario puede reordenar la lista de roles y cambiar el carácter obligatorio de un rol a opcional o viceversa.

b) Directamente sobre el resumen, el usuario puede cambiar el orden de las secciones, contraer secciones obligatorios (las convierte en opcionales) o expandir secciones opcionales (las convierte en obligatorias).

3.5 Intereses

Si es importante conocer el conocimiento, la experiencia y los gustos del usuario, no lo es menos, saber cuál es el objetivo que desea alcanzar durante su navegación en el sistema. Es decir, cuáles son sus deseos y metas, en definitiva qué cosas son las que le interesan. Los intereses recogidos en el modelo de usuario son concretamente tres: **subdominio de interés**, **ítems y conceptos interesantes**, y **meta de conocimiento**.

Saber cuáles son los intereses del usuario, permite al sistema focalizar la adaptación para conseguir que el usuario alcance su finalidad, ya sea ésta lograr un determinado estado de conocimiento o simplemente comprender el contenido de una serie de ítems de información que le interesan especialmente.

3.5.1 Subdominio de interés

Cada estructura conceptual de presentación captura una parcela concreta del dominio de conocimiento representado en la estructura conceptual de memorización. Para poder "catalogar" las presentaciones, el autor debe especificar los distintos subdominios que constituyen e integran el dominio de conocimiento completo. Indicando después para cada presentación, el subdominio o subdominios de cocimiento que incluye y en qué grado.

El sistema no puede adivinar si el usuario siente predilección por algún subdominio en particular y ni mucho menos de cuál se trata. Por este motivo, por defecto, la entrada del modelo de usuario para el atributo subdominio-de-interés tiene el valor "indiferente" y no permite actualización automática. Por supuesto, el usuario puede apuntar, en cualquier momento, su interés especial por alguno de los subdominios en los que se descompone el dominio de conocimiento del sistema hipermedia.

La entrada subdominio-de-interés se usa, junto con la experiencia del usuario, para elegir la estructura de navegación que mejor se ajusta a su perfil (capítulo 14). De esta forma, el sistema es capaz de proporcionar al usuario una estructura conceptual de aprendizaje, EC_A^i, que le permita instruirse en los conceptos del subdominio que le interesa. Esto es, que la presentación incluida en EC_A^i capture en cierto grado el subdominio deseado.

3.5.2 Ítems y conceptos interesantes

Dentro de un mismo subdominio de conocimiento, el usuario puede tener más empeño en conocer unos ítems que otros. Del mismo modo, uno o varios conceptos pueden llamar su atención por encima de los demás. Por lo tanto, es conveniente que el sistema

permita al usuario establecer sus intereses también de esta forma. Es decir, que otorgue al usuario la capacidad de marcar determinados ítems y conceptos como interesantes.

Def 12.4 [Ítem interesante / Concepto interesante] Un ítem o concepto interesante es simplemente aquél por cuyo conocimiento el usuario tiene una especial predilección. Siempre y cuando sea marcado como tal en su modelo de usuario. El conjunto de todos los ítems y conceptos interesantes es denominado *Ints*.

Esta información se utiliza en la navegación por conocimiento (capítulo 18) para resaltar los ítems accesibles que contienen información interesante para el usuario, recordándole así su interés en visitar éstos. También se resaltan aquellos ítems que el usuario debe conocer mejor para que un ítem interesante se convierta en accesible (capítulo 19, Ítems deseables). En definitiva, se trata de sugerir al usuario la visita de información muy relacionada con sus intereses.

En un primer momento, ningún ítem ni concepto se etiqueta como interesante. Debe ser el usuario el que establezca, explícitamente, un interés especial por un determinado subconjunto de éstos. Sin embargo, el sistema es capaz de inferir automáticamente un conjunto de ítems interesantes a partir de las preferencias-sobre-ítems definidas por el usuario (en caso de que las hubiera, claro está). Esta actualización automática sólo se realiza si el usuario la solicita.

Para ello, el sistema calcula el grado en que cada ítem i_j existente en la EC_M (y por lo tanto incluido en el modelo de usuario) se ajusta a las preferencias indicadas por el usuario. En concreto, para cada una de las seis características recogidas en la tabla 2, se obtiene un número real comprendido en el intervalo [-1, 1] que indica el grado en que el ítem considerado se ajusta o separa de las preferencias establecidas para dicha característica.

Notamos **indicador(i_j, característica^i)** al valor que indica lo adecuado que es el ítem i_j respecto a las preferencias establecidas por el usuario para la característica^i. El valor 1, esto es, indicador(i_j, característica^i) = 1, indica que el ítem i_j presenta para la característica^i el valor que más gusta al usuario. Por el contrario, cuando indicador(i_j, característica^i) = -1 significa que el valor de la característica^i en el ítem i_j es precisamente el que menos gusta al usuario. Finalmente, el valor 0 indica que en las preferencias sobre la característica^i el usuario no ha dicho nada (ni positivo ni negativo) referente al valor que para esta característica presenta i_j.

El valor del indicador(i_j, característica^i) se calcula aplicando el siguiente procedimiento:

Paso 1. Se obtiene el valor, **v**, que para la característica considerada, **característica^i**, tiene el ítem evaluado, **i_j**, y se busca **v** en la lista de preferencias indicadas por el usuario para dicha característica, **preferencias(característica^i)**.

Paso 2. *Si* el valor **v** no se encuentra en la lista **preferencias(característica^i)** *Entonces*

indicador(i_j, característica^i) = 0.

Paso 3. *En otro caso*, siendo **posV** la posición del valor **v** en la lista de preferencias(característica^i) y **posF** la posición del último elemento de dicha lista *Hacer*

Si el valor **v** es positivo (marca SI) *Entonces*

$$indicador(i_j, característica^i) = \frac{posF - (posV - 1)}{posF}$$

Si el valor **v** es negativo (marca NO) *Entonces*

$$indicador(i_j, característica^i) = -\left[\frac{posF - (posV - 1)}{posF}\right]$$

Algoritmo 1. Obtener indicador(i_j, característica^i)

Una vez obtenido el indicador del ítem i_j para cada característica^i con $i \in A$, se procede a obtener un indicador total que refleje lo bueno o lo malo que es ese ítem de acuerdo a las preferencias del usuario. Este indicador es denominado **indicador(i_j)** y se calcula sumando todos los indicadores parciales, indicador(i_j, característica^i), cada uno de ellos ponderado con el peso que el usuario le ha asignado a la característica correspondiente, esto es, peso(característica^i). La ecuación 6 muestra el cálculo del indicador total para un ítem i_j.

$$\text{indicador}(i_j) = \sum_{i=1}^{6} peso(característica^i) \times indicador(i_j, característica^i) \qquad (6)$$

Adviértase que la distribución de pesos que se utiliza en la ecuación 6, es aquella que considera el peso de preferencias-sobre-ítems igual a 1. Y que por tanto, el peso de una característica, peso(característica^i), corresponde al peso asignado por el usuario, sin las modificaciones que el sistema realiza para ajustarlo cuando el peso de ruta-corta es distinto de cero. Esto es se usa peso(característica^i) y no peso'(característica^i).

Puesto que $\sum_i$ peso(característica^i) = 1, se deduce de la ecuación 6, que el indicador total de un ítem, indicador(i_j), es de nuevo, un número real comprendido en el intervalo [-1, 1]. La interpretación de este indicador es la misma explicada anteriormente para los indicadores parciales. Esto es, 1 significa que el ítem i_j se ajusta completamente a los gustos del usuario y –1 que está lo más alejado posible de éstos. Diremos que las características de un ítem i_j son más del agrado del usuario que las de un ítem i_k, si se satisface que: indicador(i_j) > indicador(i_k).

Una vez obtenido el indicador total de cada ítem, el sistema infiere como intereses del usuario aquellos ítems cuyo indicador total supera un determinado umbral positivo. Es decir, aquellos cuyas características se ajustan en un grado suficiente a las preferencias del usuario. Por defecto, el umbral empleado es 0'5, pero en cada inferencia el sistema consulta al usuario el umbral que desea utilizar para identificar el conjunto de ítems interesantes a partir de sus preferencias.

A continuación, en la figura 4, se muestra un ejemplo donde se calcula, a partir de las preferencias establecidas por el usuario en la figura 2, el indicador total de dos ítems diferentes: ítem I1 e ítem I2.

Para cada ítem se desarrolla el cálculo de su indicador parcial para cada una de las seis características ("idioma", "medio", "rol", "dificultad", "fecha" y "autor"), y a partir de éstos se obtiene su indicador total. Los indicadores totales, indicador(I1) e indicador(I2), ponen de manifiesto que el ítem I1 se ajusta mejor a los gustos del usuario que el ítem I2.

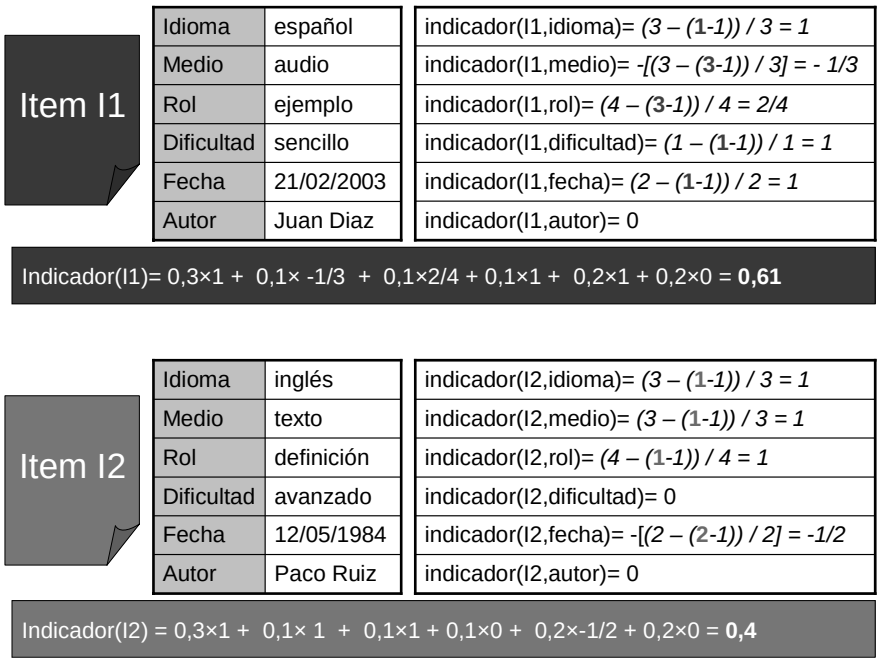


Figura 4: Inferencia automática de ítems interesantes

Observe que el indicador parcial obtenido para una característica es mayor en valor absoluto, cuando el valor que presenta el ítem en dicha característica aparece al principio de la lista de preferencias y menor cuanto más abajo se encuentra en la lista. Por ejemplo, respecto al atributo "rol", el indicador del ítem I2 es 1 porque el valor "definición" aparece marcado positivamente en la primera posición de la lista. Mientras que para el ítem I1 es 2/4 porque su rol, "ejemplo" aparece, también positivo, pero en la tercera posición de esa lista de preferencias.

Observe también que el indicador parcial obtenido para una característica es negativo cuando el valor que tiene esa característica en el ítem aparece marcado con un *NO* en la lista de preferencias del usuario. Así, el indicador para "fecha" en el ítem I2 es -1/2, ya que su fecha de creación es anterior a 1998, circunstancia marcada negativamente en la segunda posición de la lista de preferencias establecida para esa característica.

Retomando el cálculo automático de ítems interesantes, si consideramos el umbral de 0,5 por defecto: El ítem I1 se considera interesante porque su indicador total (0,61) supera el umbral mínimo establecido, mientras que el ítem I2 no lo es ya que su indicador (0,4) queda debajo de este umbral.

3.5.3 Meta de conocimiento

El sistema también ofrece al usuario la oportunidad de especificar una meta de conocimiento sobre un determinado conjunto de ítems y/o conceptos. De esta forma, el usuario no sólo determina en qué ítems o conceptos está especialmente interesado, sino también el grado de conocimiento que desea alcanzar sobre cada uno de ellos. El autor, en este caso ejerciendo de tutor, podría predefinir una serie de metas de conocimiento, que después los usuarios adoptasen según su capacidad y finalidad en el uso del sistema. Sin embargo, lo ideal es que cada usuario especifique su propia meta, ya que de esta forma su predisposición para alcanzarla será mayor.

Def 12.5 [Meta de conocimiento] Una meta de conocimiento, denominada M, es un conjunto de parejas formadas por un ítem o concepto y un grado de conocimiento. Esto es, $M = \{m_i, m_j, ..., m_k\}$. Cada una de las parejas incluida en la meta, $m_j \in M$, es una submeta que establece el grado de conocimiento mínimo (expresado mediante una de las etiquetas semánticas disponibles) que el usuario desea lograr sobre el ítem o concepto involucrado.

Def 12.6 [Submeta de conocimiento] Una submeta de conocimiento m_j se define sobre un elemento e_j y tiene la forma: (e_j, et_{SEM}^j). La submeta m_j es satisfecha cuando el grado de conocimiento que el usuario posee sobre el elemento e_j es mayor o igual que el indicado en et_{SEM}^j. Esto es, m_j se satisface sólo si $K(e_j) \geq$ valor-numérico(et_{SEM}^j). El elemento e_j puede ser un ítem, i_j, o un concepto, c_j.

En una misma meta de conocimiento, no puede haber dos submetas asociadas al mismo elemento. Además, para que se satisfaga la meta completa deben satisfacerse todas y cada una de las submetas que la componen (véase ecuación 7). Obviamente, para evaluar una meta de conocimiento, el sistema debe consultar las entradas del modelo de usuario que almacenan el grado de conocimiento poseído por éste sobre cada uno de los ítems y conceptos incluidos en la meta.

$$M = \{(e_j, et_{SEM}^j), ..., (e_k, et_{SEM}^k)\} \qquad (7)$$

$$M \text{ se satisface si } K(e_j) \geq \text{valor-numérico}(et_{SEM}^j) \wedge ... \wedge K(e_k) \geq \text{valor-numérico}(et_{SEM}^k)$$

Una meta de conocimiento no tiene porqué incluir una submeta de conocimiento para cada ítem y concepto del modelo de usuario. Se trata pues del deseo de alcanzar un estado de conocimiento parcial.

Def 12.7 [Ítem meta / Concepto meta] Un ítem o concepto meta es todo aquél para el que existe una submeta de conocimiento en M. Es decir, i_j es un ítem meta si $\exists m_j \in M$, donde $m_j = (i_j, et_{SEM}^j)$ y c_j es un concepto meta si $\exists m_j \in M$, donde $m_j = (c_j, et_{SEM}^j)$.

De acuerdo a la meta de conocimiento, el sistema es capaz de generar una *ruta de navegación personalizada* (capítulo 20) que, tras ser recorrida por el usuario, conduzca a éste a un estado de conocimiento en el cual la meta se satisface completamente. Además, en la navegación por conocimiento (capítulo 18), el sistema lleva a cabo para los ítems meta el mismo proceso de adaptación explicado para los ítems interesantes (capítulo 19, *Ítems deseables*). En ambos casos, aunque de distinto modo, lo que se hace es aconsejar al usuario para lograr el conocimiento deseado sobre cada ítem o concepto meta: la ruta guiada es un consejo global, mientras que la anotación de ítems deseables supone un consejo local.

Por defecto, el sistema no construye ninguna meta de conocimiento para el usuario, apostando por que sea el propio usuario quién establezca explícitamente su meta. Para ello, sólo tiene que escribir junto a cada ítem y concepto que le apetezca el grado de conocimiento que desea alcanzar. Sin embargo, si el usuario previamente ha definido sus preferencias y/o intereses, el sistema puede construir automáticamente una meta de conocimiento basada en éstos.

Esta actualización automática sólo tiene lugar si el usuario así lo requiere. El sistema construye la meta de manera muy sencilla. Inicialmente M es vacío, $M = \varnothing$. Después, para cada elemento interesante (ítem o concepto) se incluye en M una submeta, donde

se exige un conocimiento "total" sobre dicho elemento. Es decir, si e_j (i_j o c_j) es interesante, $e_j \in$ Ints, entonces $M = M \cup \{m_j\}$, donde $m_j = (e_j$, "total"). El conjunto de ítems interesantes utilizado puede ser el especificado por el usuario o el inferido por el sistema a partir de sus preferencias (sección 3.5.2), en el caso de que el usuario no haya especificado intereses.

4. OBTENCIÓN Y UTILIDAD DE LOS ELEMENTOS DEL MU

En esta sección se resume la información almacenada en el modelo de usuario, que se ha descrito ampliamente a lo largo de la sección anterior. Utilizando una estructura tabular, para cada entrada del modelo de usuario se indica brevemente su significado, utilización, inicialización y actualización.

De nuevo, las distintas entradas del modelo de usuario aparecen agrupadas en las cinco categorías enumeradas inicialmente: datos personales, conocimiento, experiencia, preferencias e intereses. La categoría conocimiento se desglosa en dos subcategorías, según se trate del conocimiento acerca de los conceptos o acerca de los ítems.

El objetivo de la tabla es recapitular el gran volumen de información mantenido en el modelo de usuario. Al mismo tiempo que se hace hincapié en las capacidades adaptativas que este amplio conocimiento acerca del usuario proporciona al sistema. Por último se indica quién y cómo se gestiona el modelo de usuario, tanto en su inicialización como durante la actualización de la información mantenida. En resumen, la tabla 3, nos sirve para abstraernos de los detalles y alcanzar una visión de conjunto de la información recogida en el modelo de usuario.

Tabla 3: Información del modelo de usuario

Entrada	Significado	Utilidad	Inicialización	Actualización
DATOS PERSONALES: clave, nombre, apellidos, sexo, edad y ocupación				
Datos personales	Información personal del usuario	Personalizar la comunicación y restringir el control sobre la adaptación	El usuario	El usuario
CONOCIMIENTO (ÍTEMS)				
Nº visitas	Número de accesos al contenido del ítem	Adaptar la navegación dependiendo de los ítems ya visitados e inferir experiencia de navegación	Automática (0)	Automática (contabilización de accesos)
Grado de conocimiento	Nivel de conocimiento del usuario sobre el ítem	Adaptar la navegación al conocimiento que posee el usuario	Automática ("nulo")	Automática (reglas de actualización)
CONOCIMIENTO (CONCEPTOS)				
Nº visitas	Nº de accesos al resumen del concepto	Información y estadísticas	Automática (0)	Automática (contabilización de accesos)

Grado de conocimiento	Conocimiento del usuario sobre el concepto	Informar al usuario de su conocimiento sobre conceptos e inferir experiencia en la materia	Automática ("nulo")	Automática (reglas de peso)
EXPERIENCIA				
Experiencia en la materia	Conocimiento general sobre el dominio conceptual del SH	Elegir la estructura de navegación adecuada (EC_A)	El usuario	El usuario Automática (solicitada por el usuario)
Experiencia de navegación	Práctica del usuario en el uso de SH y similares	Elegir la EC_A	El usuario	El usuario Automática (solicitada por el usuario)
PREFERENCIAS				
Preferencias sobre ítems	Características de un ítem que gustan y/o disgustan al usuario. Ponderación de las características	Generar rutas guiadas personalizadas. Inferir ítems interesantes	El usuario Automática (distribución de pesos uniforme)	El usuario
Preferencia de ruta corta	Grado con que el usuario prefiere rutas guiadas de corta longitud	Generar rutas guiadas personalizadas	Automática (0)	El usuario
Preferencia para resúmenes	Estructura de composición para el resumen de un concepto	Adaptar los resúmenes obtenidos en la navegación por conceptos	El autor	El usuario
INTERESES				
Subdominio de interés	Subdominio de conocimiento que el usuario desea navegar	Elegir la EC_A	El usuario Automática ("indiferente")	El usuario
Ítems y conceptos interesantes	Ítems y conceptos que el usuario desea aprender especialmente	Adaptar la navegación a los intereses del usuario	El usuario Automática (inferir desde preferencias)	El usuario Automática (inferir desde las preferencias)
Meta de conocimiento	Estado de conocimiento parcial que el usuario quiere alcanzar	Generar rutas guiadas y adaptar la navegación a los ítems y conceptos meta del usuario	El usuario Automática (inferir desde los intereses)	El usuario Automática (inferir desde los intereses)

5. ESQUEMA E INSTANCIA DEL MU

Respecto al modelo de usuario es importante separar dos aspectos diferentes que a menudo es fácil confundir o entremezclar. Se trata del **esquema** y la **instancia** del modelo de usuario.

Def 12.8 [Esquema del MU] El esquema del modelo de usuario es el diseño global de éste, es decir la forma en que se representan y organizan los distintos elementos que lo componen. En definitiva la *estructura* del modelo. Existe un esquema del modelo de usuario *único para cada sistema hipermedia*. Los cambios en el esquema están motivados por las modificaciones que el autor realiza en la estructura conceptual de memorización de dicho sistema hipermedia.

Def 12.9 [Instancia del MU] Una instancia del modelo de usuario es la *colección de información almacenada en el modelo para un usuario concreto*. En un sistema hipermedia existen *tantas instancias* del modelo de usuario *como usuarios diferentes* se hayan registrado en el mismo. Una instancia del modelo de usuario es una copia del esquema donde en cada entrada se rellenan los datos de ese usuario particular. Por lo tanto, una instancia cambia con tanta frecuencia como el usuario al que representa.

5.1 Esquema del modelo de usuario

Para organizar mejor la información mantenida en el modelo de usuario, su esquema se estructura o divide en entradas. Cada **entrada del modelo de usuario** está destinada a almacenar una información concreta acerca del usuario, por ejemplo su nombre o su grado de experiencia en la materia. Para representar una entrada podemos usar una n-tupla similar a la siguiente: [elemento1, elemento2,..., elementoN], cuyo número de elementos depende de la entrada en cuestión. Entre los elementos de una entrada podemos encontrar dos tipos diferentes:

a) **Elementos constantes** que se utilizan para concretar el tipo de información almacenada en la entrada. Son de la forma, atributo-constante = valor, donde el valor asociado al atributo no cambia casi nunca. Y cuando lo hace se debe a modificaciones estructurales realizadas por el autor en el sistema hipermedia.

b) **Elementos variables** que establecen cuál es el valor del usuario para el atributo indicado. Son de la forma, atributo-variable = valor, donde el valor cambia a menudo, como consecuencia de una actualización explícita del usuario o automática a medida que éste navega.

Tanto los atributos constantes como variables de los elementos de una entrada pertenecen al esquema del modelo de usuario, puesto que sirven para definir su estructura. La utilidad del valor de un elemento constante es también estructural por lo que forma parte del esquema. Sin embargo, el valor de un elemento variable supone una información concreta sobre un usuario, siendo por tanto propio de la instancia del modelo.

Por ejemplo en la entrada [categoría = datos_personales, edad = “ ”], el elemento categoría = datos_personales es constante e identifica el tipo de la entrada, mientras que edad = “ ” es un elemento variable cuyo valor no compete al esquema del modelo sino a su instancias.

El esquema del modelo de usuario no es igual para todos los sistemas hipermedia. Esto se debe a que dicha estructura depende, en parte, de la información proporcionada en el sistema y de los conceptos involucrados. Por tanto, es necesario distinguir dentro del esquema del MU:

a) una parte **común** a todos los sistemas hipermedia y

b) otra parte que **depende del dominio conceptual y de información** de cada sistema.

Así, dadas dos estructuras conceptuales de memorización diferentes, EC_M^1 y EC_M^2, la estructura de los modelos de usuario construidos para cada una de ellas es idéntica en un conjunto de entradas, pero distinta en otro. Concretamente, la parte común al esquema de cualquier modelo, esto es $\forall_i EC_M^i$, está formada por el conjunto de entradas que se muestran en la tabla 4. Como puede verse, salvo la última entrada, destinada al subdominio de interés del usuario, el resto proporciona la estructura necesaria para almacenar información acerca de los datos personales, preferencias y experiencia del usuario.

Tabla 4: Entradas existentes en todos los MU

[categoría = datos_personales, nombre = “ ”]
[categoría = datos_personales, apellidos = “ ”]
[categoría = datos_personales, edad = “ ”]
[categoría = datos_personales, sexo = “ ”]
[categoría = datos_personales, ocupación = “ ”]
[categoría = datos_personales, clave = “ ”]
[categoría = experiencia, experiencia-materia = “ ”]
[categoría = experiencia, experiencia-navegación = “ ”]
[categoría = preferencias, atributo = preferencia-idioma, peso = “ ”, lista-gustos = “ ”]
[categoría = preferencias, atributo = preferencia-autor, peso = “ ”, lista-gustos = “ ”]
[categoría = preferencias, atributo = preferencia-fecha, peso = “ ”, lista-gustos = “ ”]
[categoría = preferencias, atributo = preferencia-rol, peso = “ ”, lista-gustos = “ ”]
[categoría = preferencias, atributo = preferencia-medio, peso = “ ”, lista-gustos = “ ”]
[categoría = preferencias, atributo = preferencia-dificultad, peso = “ ”, lista-gustos = “ ”]
[categoría = preferencias, atributo = preferencia-ruta-corta, peso = “ ”]
[categoría = preferencias, atributo = preferencia-resúmenes, lista-roles = “ ”]
[categoría = intereses, subdominio-de-interés = “ ”]

La parte del esquema del modelo de usuario que es distinta para cada estructura conceptual de memorización está compuesta por las entradas asociadas a sus ítems y conceptos. Como puede verse en la tabla 5, existe una entrada para cada ítem y concepto existente en una EC_M^i. Dicha entrada recoge toda la información relativa a éste: identificador, nombre, número de visitas y grado de conocimiento. Contemplando, también, si el ítem o concepto ha sido marcado como interesante por el usuario y la meta de conocimiento definida sobre éste.

Tabla 5: Entradas existentes en el $MU(EC_M^i)$

$\forall c_k \in C(EC_M^i)$: [categoría = conocimiento_e_intereses, tipo = concepto, identificador = c_k, nombre = c_k, visitas = " ", K = " ", interesante = " ", meta = " "]
$\forall i_j \in I(EC_M^i)$: [categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_j, nombre = nombre(i_j), visitas = " ", K = " ", interesante = " ", meta = " "]

En definitiva la parte del esquema del modelo de usuario que varía de un sistema a otro es el conjunto de entradas destinadas a reflejar el conocimiento del usuario sobre los ítems y conceptos disponibles, además de su interés especial por algunos de éstos.

Supongamos un sistema hipermedia cuya EC_M presenta tres conceptos con los nombres *Objeto*, *Método* y *Herencia*, y tres ítems etiquetados como *método-suma*, *método-instancia* y *herencia-en-java* con identificadores i_1, i_2 e i_3 respectivamente.

La tabla 6 muestra las entradas específicas del esquema de su modelo de usuario.

Tabla 6: Entradas específicas para el esquema del ejemplo

[categoría = conocimiento_e_intereses, tipo = concepto, identificador = Objeto, nombre = Objeto, visitas = " ", K = " ", interesante = " ", meta = " "]
[categoría = conocimiento_e_intereses, tipo = concepto, identificador = Método, nombre = Método, visitas = " ", K = " ", interesante = " ", meta = " "]
[categoría = conocimiento_e_intereses, tipo = concepto, identificador = Herencia, nombre = Herencia, visitas = " ", K = " ", interesante = " ", meta = " "]
[categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_1, nombre = método-suma, visitas = " ", K = " ", interesante = " ", meta = " "]
[categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_2, nombre = método-instancia, visitas = " ", K = " ", interesante = " ", meta = " "]
[categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_3, nombre = herencia-en-java, visitas = " ", K = " ", interesante = " ", meta = " "]

5.2 Instancia del modelo de usuario

En las tablas 4 y 5 se han representado las entradas que van a existir en el esquema de todo modelo de usuario. Se ha asignado el valor vacío, representado mediante comillas dobles, " ", a todos los atributos variables. El valor de cada uno de estos atributos será diferente para cada usuario del sistema, con lo cual no debe ser rellenado en el esquema del modelo de usuario sino en cada una de sus instancias. Consistiendo, precisamente la instanciación de un modelo de usuario, en establecer el valor concreto que posee ese usuario respecto a cada atributo-variable existente en el esquema.

Son por tanto, las instancias del modelo de usuario, no el esquema, las que sufren los procesos de inicialización y actualización explicados en las secciones anteriores. Esto es, es el valor de los atributos variables del modelo lo que cambia a medida que el usuario al que representa hace uso del sistema hipermedia. La actualización de un atributo-variable en una instancia, puede producirse, bien porque el usuario proporciona de forma directa un nuevo valor para él, o bien porque el sistema obtiene automáticamente dicho valor, bajo petición o de forma autónoma.

Continuando con el ejemplo anterior, en la tabla 7 se muestra la instanciación del esquema definido en las tablas 4 y 6 para un usuario ficticio de nombre Juan. Se han mantenido las comillas dobles para que le sea más fácil al lector identificar el valor, dado en la instanciación, a cada uno de los atributos variables del esquema. Dicho valor aparece también resaltado en negrita.

Tabla 7: Instanciación del MU para Juan

[categoría = datos_personales, nombre = "**Juan**"]
[categoría = datos_personales, apellidos = "**Gil Arias**"]
[categoría = datos_personales, edad = "**25**"]
[categoría = datos_personales, sexo = "**Hombre**"]
[categoría = datos_personales, ocupación = "**Estudiante**"]
[categoría = datos_personales, clave = "************"]
[categoría = experiencia, experiencia-materia = "**1**"]
[categoría = experiencia, experiencia-navegación = "**3**"]
[categoría = preferencias, atributo = preferencia-idioma, peso = "**0,3**", lista-gustos = "**{(sí, español, inglés), (no, ruso), (sí, italiano)}**"]
[categoría = preferencias, atributo = preferencia-autor, peso = "**0,2**", lista-gustos = "**{}**"]
[categoría = preferencias, atributo = preferencia-fecha, peso = "**0,2**", lista-gustos = "**{}**"]
[categoría = preferencias, atributo = preferencia-rol, peso = "**0,1**", lista-gustos = "**{}**"]
[categoría = preferencias, atributo = preferencia-medio, peso = "**0,1**", lista-gustos = "**{}**"]
[categoría = preferencias, atributo = preferencia-dificultad, peso = "**0,1**", lista-gustos = "**{}**"]
[categoría = preferencias, atributo = preferencia-ruta-corta, peso = "**0**"]
[categoría = preferencias, atributo = preferencia-resúmenes, lista-roles = "**{(introducción, obligatorio), (definición, obligatorio), (ejemplo, obligatorio), (aclaración, opcional), ...}**"]
[categoría = intereses, subdominio-de-interés = "**indiferente**"]
[categoría = conocimiento_e_intereses, tipo = concepto, identificador = Objeto, nombre = Objeto, visitas = "**0**", K = "**0**", interesante = "**Sí**", meta = "**0**"]
[categoría = conocimiento_e_intereses, tipo = concepto, identificador = Método, nombre = Método, visitas = "**1**", K = "**2**", interesante = "**No**", meta = "**0**"]
[categoría = conocimiento_e_intereses, tipo = concepto, identificador = Herencia, nombre = Herencia, visitas = "**2**", K = "**3**", interesante = "**No**", meta = "**0**"]
[categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_1, nombre = método-suma, visitas = "**1**", K = "**1**", interesante = "**Sí**", meta = "**3**"]
[categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_2, nombre = método-instancia, visitas = "**0**", K = "**0**", interesante = "**Sí**", meta = "**0**"]
[categoría = conocimiento_e_intereses, tipo = ítem, identificador = i_3, nombre = herencia-en-java, visitas = "**5**", K = "**4**", interesante = "**No**", meta = "**0**"]

Observe como cada atributo-variable ha sido instanciado con un valor del tipo esperado. Es decir, tal y como se especificó en el apartado correspondiente de la sección 3, la edad debe ser un número entero, el conocimiento, la meta y la experiencia son grados de conocimiento en el rango de enteros [0, 4], los pesos de las preferencias son números

reales entre 0 y 1, la estructura de un resumen es una lista ordenada de roles indicando para cada uno su carácter opcional u obligatorio, etc.

6. INTERACCIÓN DEL USUARIO CON EL MU

A continuación, en la figura 5, se propone una posible interfaz gráfica donde el usuario puede examinar la información almacenada en el modelo que sobre él mantiene el sistema, y tras esta revisión, modificar los datos que considere incorrectos o actualizar aquellos que estén obsoletos.

El diseño propuesto para la interfaz del modelo de usuario no es determinante sino puramente orientativo. Lo importante no es la imagen elegida para un botón, ni el nombre dado a una etiqueta. Lo realmente importante, y el objetivo de esta sección, es definir gráficamente cuál es la estructura que debe tener el modelo de usuario independientemente de cómo se implemente después. Indicando también la forma en que el usuario puede interaccionar con la interfaz propuesta para consultar y modificar la información del modelo.

La interfaz muestra una instancia del modelo de usuario, en este caso el precisado para Juan en la tabla 7. Es conveniente resaltar que el usuario interacciona en todo momento con su instancia del modelo de usuario, no con el esquema global del mismo. Por lo que únicamente puede cambiar el valor de los atributos-variables.

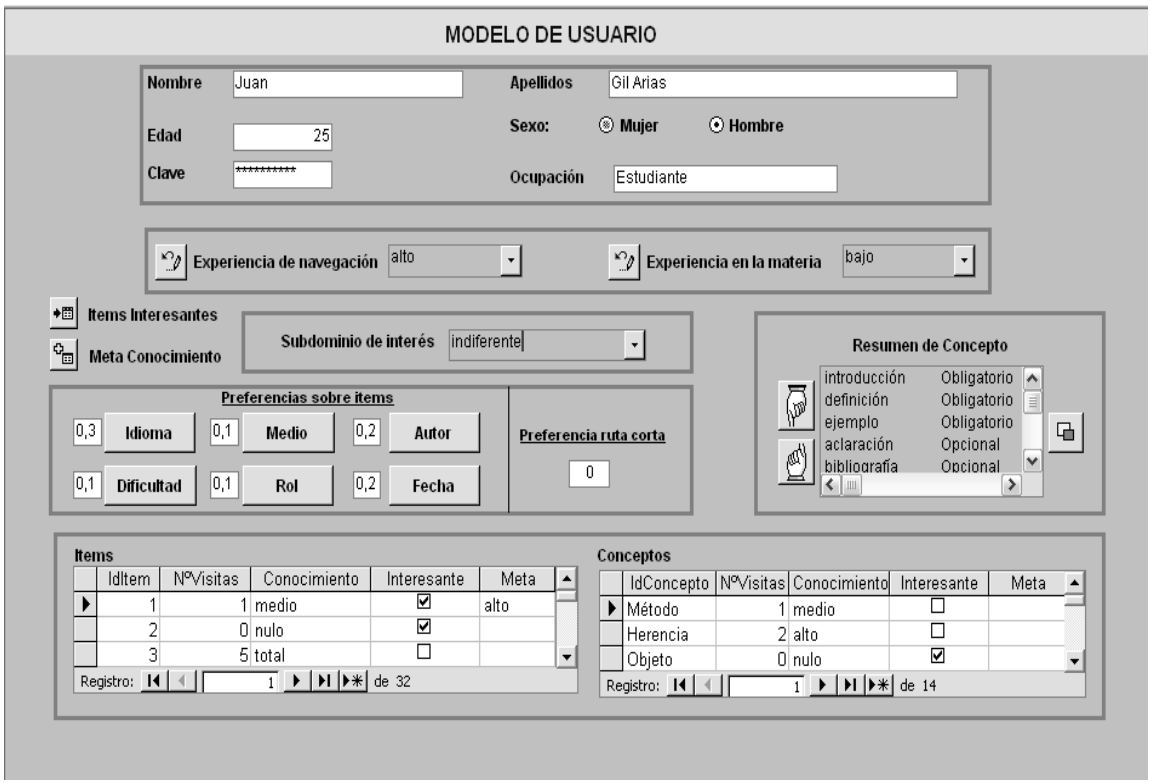


Figura 5: Modelo de usuario

Como puede observarse en la figura 5, las entradas del modelo de usuario aparecen agrupadas en distintas secciones que se corresponden más o menos con las cinco

categorías existentes. En el formulario, la primera sección que aparece está destinada a los **datos personales** del usuario: nombre, edad, sexo, clave,...

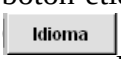
Debajo de la información personal, existe un apartado designado para recoger la **experiencia** del usuario tanto de navegación como en la materia. Para ambos tipos de experiencia, el usuario puede seleccionar una de las cinco etiquetas semánticas de la lista desplegable que se le proporciona (figura 6) o solicitar una inferencia automática pulsando el botón  que aparece junto a él. Dichas inferencias son realizadas por el sistema tal y como se explicó en las secciones 3.3.1 y 3.3.2.

Figura 6: Grados de experiencia

Debajo de la experiencia, aparecen dos botones, no enmarcados, que el usuario debe usar si desea que el sistema deduzca automáticamente un subconjunto de ítems interesantes  y/o una meta de conocimiento  a partir de las preferencias (o intereses para el caso de la meta) que el mismo ha establecido sobre los ítems. El procedimiento seguido en cada caso es descrito en las secciones 3.5.2 y 3.5.3 respectivamente.

Al lado de estos botones de actualización automática, existe una zona donde el usuario puede especificar el **subdominio de conocimiento** en el que está más interesado. Para ello sólo tiene que seleccionar uno de todos los subdominios que se le muestran en la lista despegable. En dicha lista, junto a cada subdominio se incluye una breve descripción introducida por el autor acerca del mismo, que informa al usuario sobre el ámbito de éste. Si el usuario no tiene ningún interés especial por ningún subdominio, selecciona el valor por defecto “indiferente”.

Debajo de las dos secciones anteriores, se encuentra una nueva sección dedicada a recoger las **preferencias del usuario sobre los ítems** del sistema hipermedia. Junto a cada característica (idioma, rol, dificultad, medio,...) aparece un pequeño recuadro con el peso asociado a ésta. Inicialmente el peso es 1/6 para todas. Pero si el usuario quiere primar unas características sobre otras, puede escribir en los recuadros un valor diferente. Si la distribución de pesos resultante no suma uno, el sistema avisa al usuario y le ofrece la posibilidad de automáticamente solucionar el error (ver sección 3.4.1).

Para establecer los gustos respecto a un atributo, el usuario sólo tiene que pulsar el botón etiquetado con el nombre de la característica en cuestión. Así, al pulsar el botón 

se le proporciona un subformulario donde puede especificar los valores que le gustan y los que le disgustan para ese atributo y en qué orden.

En la figura 7 se muestra la ventana donde el usuario ha indicado sus preferencias respecto al idioma con el que se expresa la información contenida en los ítems. Para ello no tiene que teclear ningún valor. Le basta con elegir uno de todos los valores posibles que se facilitan en una lista desplegable. De esta forma, se evita que el usuario

establezca preferencias acerca de idiomas que no son utilizados actualmente en ningún ítem del sistema.

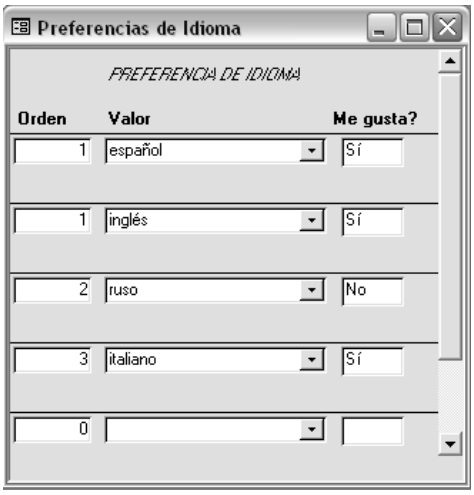


Figura 7: Preferencias de idioma

Junto a las preferencias sobre ítems, aparece un apartado donde el usuario puede reflejar su inclinación por la generación de rutas de navegación cortas (0 en el ejemplo). Cuando el valor elegido es mayor que 0 el sistema calcula internamente la nueva distribución de pesos para las preferencias-sobre-ítems tal y como se explicó en 3.4.2.

A la derecha de la sección dedicada al **peso de ruta-corta** se sitúa un cuadro dedicado a personalizar la **estructura de composición de los resúmenes** que se van a generar durante la navegación por conceptos (capítulo 16). Se trata de una lista que muestra el orden en el que deben aparecer los distintos roles en el resumen de un concepto. Además se indica para cada rol si su inclusión en el resumen es obligatoria u opcional. El usuario puede reordenar los roles pulsando los botones   para adelantar o posponer la aparición del rol seleccionado. Además permite cambiar de opcional a obligatorio o de obligatorio a opcional el carácter del rol seleccionado, usando sin más el botón  .

Abajo, la interfaz incluye una tabla con todos los **ítems** del sistema hipermedia (32 en el ejemplo). Existe una fila para cada ítem i_j. El primer campo de la fila muestra el *identificador* del ítem. El segundo campo muestra el número de veces que el usuario ha visitado el ítem, esto es *visitas(i_j)*. Y el tercer campo refleja el grado de conocimiento que el usuario posee actualmente sobre dicho ítem, para lo cual proporciona la etiqueta semántica correspondiente a *K(i_j)*.

Mientras que los tres primeros campos de la tabla no debieran ser modificados explícitamente, si es deseable que esto ocurra con los dos últimos, que el usuario debe completar para especificar sus intereses, salvo, claro está, que solicite una inferencia automática. Concretamente el penúltimo campo de la fila de un ítem i_j presenta una casilla que el usuario debe marcar para convertir a i_j en un ítem *interesante*. De modo análogo, para convertir a i_j en un ítem *meta* el usuario escribe en el último campo de su fila el grado de conocimiento mínimo que desea alcanzar sobre él.

Al lado de esta tabla, aparece otra similar, mostrando información acerca de los **conceptos** del sistema hipermedia (14 en el ejemplo). En dicha tabla existe nuevamente una fila para cada concepto, que exhibe: 1) su identificador, esto su nombre, 2) el

número de veces que se ha solicitado un resumen del concepto, 3) el grado de conocimiento alcanzado, 4) si es o no un concepto interesante para el usuario y 5) alguna meta de conocimiento asociada.

Como se comentó anteriormente, en principio, el usuario puede modificar cualquier entrada del modelo. Sin embargo, existen algunas características para las que es preferible la actualización automática, como el número de visitas y grado de conocimiento sobre ítems y conceptos. En el primer caso es obvio que el usuario puede perder la cuenta de sus visitas conforme estas van creciendo, con lo cuál una cuenta automática es mucho más fiable. Respecto al segundo, el autor establece las restricciones de acceso a los ítems y la adquisición de conocimiento dentro de un plan pedagógico conjunto. Por lo que, si el usuario falsea su adquisición de conocimiento pone en peligro su proceso de aprendizaje, visitando ítems sin estar realmente preparado o por lo menos no de acuerdo al criterio del autor. Esta situación puede hacer que las posteriores adquisiciones de conocimiento, aún realizadas según lo establecido por el autor, no sean fiables por apoyarse en la existencia de un conocimiento previo dudoso.

Por lo tanto, el sistema puede prohibir la modificación directa de estos atributos para determinados tipos de usuarios, siempre claro está, que así lo establezca el autor. Para el resto de datos, siempre se prefiere una actualización explícita del usuario.

7. ACCIONES EVOLUTIVAS DEL MU

Finalmente, se describen las acciones evolutivas que permiten modificar el esquema del modelo de usuario, para que en todo momento, se mantenga coherente con el conjunto de ítems y conceptos incluidos en la estructura conceptual de memorización del sistema hipermedia.

Así, ante la cuestión: ¿Qué parte del esquema del modelo de usuario es susceptible de sufrir cambios? La respuesta es inmediata: El conjunto de entradas del esquema que son específicas de cada estructura conceptual de memorización, esto es, las asociadas a los conceptos e ítems. Concretamente, los cambios que repercuten en el modelo de usuario son:

a) Añadir un ítem o concepto en la EC_M: supone crear una nueva entrada en el esquema del modelo de usuario.

b) Eliminar un ítem o concepto de la EC_M: supone eliminar la entrada correspondiente en el esquema.

c) Modificar el nombre de un ítem o concepto en la EC_M: supone modificar el valor del atributo-constante “nombre” en la entrada correspondiente del esquema. En el caso de un concepto supone modificar también el valor del atributo-constante “identificador”.

Dichos cambios, son realizados por el autor usando las acciones evolutivas existentes en el Sistema de Memorización. Entonces, ¿debe el autor preocuparse, también, de hacer evolucionar adecuadamente el esquema del modelo de usuario? La respuesta, debe ser y es categóricamente No. Es el sistema quién de forma automática construye el esquema del modelo de usuario, y es también éste quién lo modifica.

Como se explica más detalladamente en el siguiente capítulo (13, **Propagación externa del cambio**), el sistema realiza los cambios necesarios sobre el modelo de usuario para que, en todo momento, sea capaz de almacenar el conocimiento e interés del usuario sobre cada uno de los ítems y conceptos de la EC_M. En consecuencia, cuando el autor realiza los cambios indicados en a), b) y c) sobre la EC_M, el sistema ejecuta las acciones evolutivas de las tablas 8, 9 y 10, para mantener íntegra y completa la estructura del modelo de usuario.

El esquema del modelo de usuario se ve afectado por la actuación del autor, mientras que las instancias dependen de la evolución del usuario al que representan. Sin embargo, al modificar el esquema de un modelo de usuario la estructura de sus instancias debe ser también modificada. Se trata de una especie de propagación interna, que el sistema realiza de forma automática para mantener coherente la estructura de las instancias del modelo previamente creadas.

La propagación del cambio realizado en el esquema del modelo de usuario puede ejecutarse de dos formas diferentes: a) de golpe sobre todas sus instancias, o b) posponer su aplicación para cada instancia hasta la siguiente vez que el usuario al que corresponde se registre en el sistema. De esta última forma se evita propagar la modificación a instancias del modelo que quizá no vuelvan a usarse. Evidentemente, en este último caso sería necesario mantener una historia con los cambios realizados en el esquema del modelo de usuario, de forma que al iniciarse una sesión con una instancia, se propagasen a ésta los cambios estructurales realizados con fecha posterior a la de su última sesión.

7.1 Añadir una entrada al modelo de usuario

Tabla 8: Acciones evolutivas para modificar el modelo de usuario –AñadirElementoMU–

ACe[1][MU]	**AñadirElementoMU(elem, t, nomb)**
Argumentos	**elem** es el identificador del elemento, ítem o concepto, para el que se desea añadir una nueva entrada en el modelo de usuario. **t** es el tipo del elemento y determina si la nueva entrada corresponde a un ítem (t="ítem") o a un concepto (t="concepto"). **nomb** es el nombre del elemento. Puesto que el nombre de un concepto es su identificador, no es necesario rellenar este argumento cuando el tipo del elemento es "concepto". En este caso nomb = elem.
Definición	Esta acción evolutiva modifica la estructura del modelo de usuario añadiéndole una nueva entrada que puede corresponder a un ítem o concepto. Dicha entrada es también introducida en todas las instancias del modelo de usuario que han sido creadas previamente. En todas ellas, la información de la nueva entrada es inicializada convenientemente.

Precondición	Existe en la estructura conceptual de memorización actual, EC_M, un elemento del tipo indicado, *t*, cuyo identificador coincide con el que se pasa como argumento, *elem*. Es decir: Si t = "ítem" entonces elem $\in$ I(EC_M). Si t = "concepto" entonces elem $\in$ C(EC_M).
	No existe actualmente ninguna entrada en el esquema del modelo de usuario asociada a un elemento del mismo tipo con igual identificador. Esto es: entrada $\notin$ MU tal que: tipo(entrada) = t $\wedge$ identificador(entrada) = elem.
Efecto	Se crea una nueva entrada, nuevaEntrada, asociada al elemento indicado. nuevaEntrada = [categoría, tipo, identificador, nombre, visitas, K, interesante, meta], donde: categoría(nuevaEntrada) = conocimiento_e_intereses tipo(nuevaEntrada) = t identificador(nuevaEntrada) = elem nombre(nuevaEntrada) = nomb visitas(nuevaEntrada) = " " K(nuevaEntrada) = " " interesante(nuevaEntrada) = " " meta(nuevaEntrada) = " " La nueva entrada se añade al esquema del modelo de usuario. MU = MU $\cup$ {nuevaEntrada}.
Propagación interna	Todas las instancias del MU creadas en un futuro para nuevos usuarios del sistema serán construidas de acuerdo a la nueva estructura. Sin embargo, las instancias del modelo previamente creadas deben ser modificadas, añadiéndoles la nueva entrada. Esta operación puede posponerse para cada instancia hasta que el usuario al que corresponde inicie una nueva sesión en el sistema. En ambos casos, la inicialización de los atributos-variables para la nueva entrada en la instancia, es la siguiente: visitas(nuevaEntrada) = "0" K(nuevaEntrada) = "0" ("nulo") interesante(nuevaEntrada) = "No" meta(nuevaEntrada) = "0" ("nulo")
Salida	No es necesaria ninguna salida de control ya que esta acción es ejecutada únicamente por el sistema cuando se añade un nuevo ítem o concepto en la estructura conceptual de memorización. Por lo que se asegura la

	satisfacción de las precondiciones exigidas.
Ejecutada_por	Sistema.

7.2 Eliminar una entrada del modelo de usuario

Tabla 9: Acciones evolutivas para modificar el modelo de usuario –EliminarElementoMU–

ACe^2[MU]	**EliminarElementoMU(elem, t)**
Argumentos	**elem** es el identificador del elemento, ítem o concepto, para el que se desea eliminar su entrada en el modelo de usuario. **t** es el tipo del elemento y determina si la entrada a eliminar corresponde a un ítem (t="ítem") o a un concepto (t="concepto").
Definición	Esta acción evolutiva elimina una entrada en el esquema del modelo de usuario. La entrada eliminada correspondiente a un ítem o concepto. Dicha entrada es también eliminada en todas las instancias del modelo previamente existentes.
Precondición	No existe en la estructura conceptual de memorización actual un elemento con igual tipo e identificador que aquél para el que se desea borrar su entrada en el MU. Es decir: Si t = "ítem" entonces elem $\notin$ I(EC_M). Si t = "concepto" entonces elem $\notin$ C(EC_M). Así se asegura que el modelo de usuario almacena información acerca de todos lo ítems y conceptos incluidos en EC_M.
	Existe en el esquema del modelo de usuario la entrada que se desea eliminar. Esto es: entrada $\in$ MU tal que tipo(entrada) = t $\wedge$ identificador(entrada) = elem.
Efecto	Se elimina la entrada del esquema del modelo de usuario cuyo tipo e identificador coinciden con los que se pasan como argumento. entradaEliminada $\in$ MU, tal que: tipo(entradaEliminada) = t $\wedge$ identificador(entradaEliminada) = elem. MU = MU – entradaEliminada.
Propagación interna	Todas las instancias del MU creadas en un futuro para los nuevos usuarios del sistema serán construidas de acuerdo a la nueva estructura, esto es, no incluirán la entradaEliminada. Sin embargo, todas las instancias del modelo previamente creadas deben

	ser modificadas, eliminando dicha entrada. Esta operación puede posponerse para cada instancia hasta que el usuario al que corresponde inicie una nueva sesión en el sistema.
Salida	No es necesaria ninguna salida de control ya que esta acción es ejecutada únicamente por el sistema cuando se elimina un ítem o concepto de la estructura conceptual de memorización. Por lo que se asegura la satisfacción de las precondiciones exigidas.
Ejecutada_por	Sistema.

7.3 Modificar una entrada en el modelo de usuario

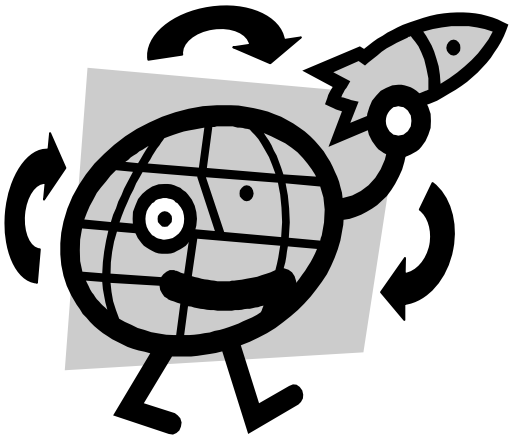
Tabla 10: Acciones evolutivas para modificar el modelo de usuario –ModificarElementoMU–

ACe^3 [MU]	**ModificarElementoMU(elem, t, nuevoNomb)**
Argumentos	**elem** es el identificador del elemento, ítem o concepto, para el que se desea modificar su entrada en el modelo de usuario. **t** es el tipo del elemento y determina si la entrada corresponde a un ítem (t="ítem") o a un concepto (t="concepto"). **nuevoNomb** es el nuevo nombre que se desea para el elemento.
Definición	Esta acción evolutiva modifica la entrada asociada a un ítem o concepto en el esquema del modelo de usuario. Concretamente sustituye el nombre actual de la entrada, nomb, por el nuevo nombre que se pasa como argumento, nuevoNomb. Como para un concepto el nombre es su identificador, también se modifica el identificador de la entrada si se trata de un concepto. En cualquier caso la modificación realizada es propagada a todas las instancias del modelo existentes previamente.
Precondición	En la estructura conceptual de memorización actual existe el elemento asociado a la entrada modificada: Si la entrada es de tipo concepto debe existir en la EC_M un concepto con el nuevo nombre. Esto es, si t = "concepto" entonces nuevoNomb $\in$ C(EC_M). Si la entrada es de tipo ítem debe existir en la EC_M un ítem con el identificador indicado. Esto es, si t = "ítem" entonces elem $\in$ I(EC_M).
	Debe existir en el esquema del modelo de usuario la entrada que se desea modificar. entradaModificada $\in$ MU tal que: tipo(entradaModificada) = t $\wedge$ identificador(entradaModificada) = elem.

Efecto	Si la entrada corresponde a un ítem se modifica sólo el nombre de la entrada. Esto es: Si t = “ítem” entonces nombre(entradaModificar) = nuevoNomb. Si la entrada corresponde a un concepto se modifica el nombre y el identificador de la entrada. Esto es: Si t = “concepto” entonces nombre(entradaModificar) = identificador(entradaModificar) = nuevoNomb.
Propagación interna	Todas las instancias del MU creadas en un futuro para nuevos usuarios del sistema serán construidas de acuerdo al esquema modificado. Sin embargo, en todas las instancias del modelo previamente creadas es necesario realizar la modificación indicada. Esta operación puede posponerse para cada instancia hasta que el usuario al que corresponde inicie una nueva sesión en el sistema.
Salida	No es necesaria ninguna salida de control ya que esta acción es ejecutada únicamente por el sistema cuando el nombre de un ítem o concepto es modificado en la estructura conceptual de memorización. Por lo que se asegura la satisfacción de las precondiciones exigidas.
Ejecutada_por	Sistema.

CAPÍTULO 13

Propagación Externa del Cambio



Resumen

Con objeto de integrar adecuadamente el Sistema de Aprendizaje dentro del modelo SEM-HP, en el presente capítulo se estudia y especifica el proceso de propagación externa que debe ejecutarse sobre el Sistema de Aprendizaje como consecuencia de un cambio acontecido en el resto de sistemas que componen el modelo. Se establece la estrecha relación entre el Sistema de Aprendizaje y los Sistemas de Memorización y Presentación, indicando para cada una de las acciones evolutivas ofrecidas en éstos la propagación necesaria sobre los elementos del primero.

Tabla de contenidos

Propagación Externa del Cambio

1. INTRODUCCIÓN

Durante la descripción formal de los elementos que constituyen el Sistema de Aprendizaje, se han definido las acciones evolutivas que permiten al autor modificar cada uno de los componentes de este sistema. Mostrando también, cómo las precondiciones y poscondiciones exigidas en cada acción permiten mantener, no sólo la consistencia del elemento modificado, sino también, la integridad del sistema completo.

Al igual que el Sistema de Aprendizaje, el resto de sistemas que integran el modelo SEM-HP, esto es, los Sistemas de Memorización, Presentación y Navegación, proporcionan un conjunto de acciones evolutivas que son ofrecidas al autor para que éste pueda realizar sobre sus respectivos componentes los cambios que considere necesarios, de una forma sencilla y consistente.

Debido a la estrecha interrelación entre los sistemas que constituyen el modelo SEM-HP, no es suficiente mantener la coherencia dentro de cada sistema de forma aislada e independiente del resto. Sino que por el contrario, se hace imprescindible un mecanismo de propagación externa del cambio, que avale la **co-evolución** consistente de los cuatro sistemas [Medina, 03d].

Existen tres fases claramente diferenciadas en la comprobación de consistencia según cuál sea el alcance de la misma. De menor a mayor alcance éstas son:

1) Garantizar la consistencia del elemento modificado.

2) Garantizar la consistencia del sistema al que pertenece el elemento modificado, propagando si es necesario el cambio a otros elementos de dicho sistema.

3) Garantizar la consistencia del conjunto de sistemas que a través de sus interrelaciones constituyen el sistema hipermedia, propagando en función de éstas los cambios producidos en uno de los sistemas a todos los sistemas con los que se relaciona.

La *consistencia a nivel de elemento* (1) es controlada mediante las **restricciones**, precondiciones y/o poscondiciones, exigidas en las acciones evolutivas.

En el Sistema de Aprendizaje, ejemplo de esto son las precondiciones que se encargan de asegurar que no existen dos predicados de actualización sobre el mismo ítem en una regla de actualización o que la suma de los pesos de una regla de peso es siempre uno. También las poscondiciones que comprueban la consistencia del conjunto de reglas de conocimiento y actualización para garantizar la alcanzabilidad de todos los ítems.

Para asegurar la *consistencia a nivel de sistema* (2) no es suficiente el uso de restricciones en las acciones evolutivas. También se requiere un mecanismo de **propagación interna** del cambio, que mantenga coherente el conjunto completo de elementos del sistema.

En el Sistema de Aprendizaje, un ejemplo de esto es el proceso de propagación automático que tiene lugar en las instancias de un modelo de usuario cuando cambia el esquema de éste.

Para mantener la *consistencia a nivel de todo el modelo* (3) es necesario un proceso automático de propagación del cambio, que "extrapole" fuera de un sistema los cambios realizados en el mismo. Se trata de un proceso de **propagación externa** que tras los cambios realizados en el sistema actual modifica en el resto de sistemas los elementos que se vean afectados. Por lo tanto, es esencial identificar para cada sistema:

a) ¿Cuáles son los cambios estructurales en este sistema que repercuten en los otros sistemas?, y

b) ¿Qué cambios en los otros sistemas repercuten en la estructura de este sistema?

En el presente capítulo planteamos y resolvemos ambas cuestiones en cuanto al Sistema de Aprendizaje se refiere. Concretamente la respuesta a la **primera pregunta** parece inmediata: Los cambios estructurales en el Sistema de Aprendizaje no pueden repercutir en el resto de sistemas, ya que éste es el último paso en el proceso de desarrollo incremental. Es decir, los elementos del Sistema de Aprendizaje se definen a partir de los elementos creados en los anteriores sistemas, no al revés (véase la ecuación 1).

$$\mathbf{EC_A^j} = (\mathbf{EC_M}, Rw, MU, \mathbf{EC_P^k}, EC_N^i, Ru^j, Rk^j) \qquad (1)$$

Es cierto que, directamente a partir de los elementos estructurales creados en el Sistema de Aprendizaje no se define ningún elemento en el resto de los sistemas. Sin embargo, es necesario recalcar los términos "directamente" y "estructurales", ya que como se explica más adelante (capítulo 21, Retroalimentación adaptativa), durante el funcionamiento del sistema hipermedia, el Sistema de Aprendizaje analiza el comportamiento navegacional de los usuarios y en base a éste sugiere modificaciones en el resto de sistemas. Después, el autor puede ejercer o no esos cambios, para adaptar sus elementos de representación al uso mayoritario que de ellos hacen los usuarios.

Responder a la **segunda pregunta**, equivale a establecer qué elementos del Sistema de Aprendizaje se definen sobre qué elementos de los otros sistemas. En este sentido, podemos determinar que el modelo de usuario (capítulo 12) y las reglas de peso (capítulo 11) se definen sobre la estructura conceptual del Sistema de Memorización, mientras que las reglas de actualización (capítulo 9) y las reglas de conocimiento (capítulo 8) se establecen sobre los ítems de una presentación concreta.

De hecho, si repasamos las acciones evolutivas proporcionadas para cada uno de estos elementos, podemos ver que contienen algunas restricciones que hacen referencia a determinados elementos de los citados sistemas.

Por ejemplo, al añadir una entrada al modelo de usuario se comprueba que corresponda a un ítem o concepto existente en la estructura conceptual de memorización.

Al añadir un término en una regla de peso se comprueba que, en la estructura conceptual de memorización, el ítem al que corresponde el nuevo término está asociado funcionalmente al concepto de la regla.

Al incluir un nuevo predicado en una regla de conocimiento o actualización se exige que el ítem al que éste se asocia aparezca en la estructura conceptual de presentación sobre la que se define la regla.

Para determinar de forma precisa los cambios que al ser ejecutados sobre algún otro sistema tienen algún tipo de repercusión en los elementos del Sistema de Aprendizaje, se ha realizado un exhaustivo análisis de las posibilidades de evolución que existen en los sistemas de Memorización, Presentación y Navegación. Obteniendo a partir de este estudio las siguientes conclusiones generales, que constituyen finalmente la respuesta a la segunda cuestión planteada:

- Puesto que el conjunto de **reglas de actualización**, $Ru(EC_A^j)$, **y de conocimiento**, $Rk(EC_A^j)$, son definidas, por defecto o por el propio autor, sobre una determinada estructura conceptual de presentación (EC_P^k), existen ciertos cambios que al ser llevados a cabo requieren que el *Sistema de Presentación* inicie un proceso de propagación sobre los citados elementos del Sistema de Aprendizaje.

- Debido a que el conjunto de **reglas de peso** se define para cada uno de los conceptos de la estructura conceptual de memorización (EC_M) y a que, en el esquema del **modelo de usuario** debe existir una entrada para cada concepto e ítem presente en dicha estructura, hay cambios sobre el *Sistema de Memorización* que obligan a modificar los mencionados componentes en el Sistema de Aprendizaje.

- Además, algunos *cambios en el Sistema de Memorización desencadenan un proceso de propagación externa sobre las presentaciones creadas a partir de éste*, lo que puede provocar, a su vez, la necesidad de nuevos cambios en las reglas de actualización y conocimiento definidas en el Sistema de Aprendizaje.

2. PROPAGACIÓN DESDE EL SISTEMA DE PRESENTACIÓN

Cuatro son las acciones evolutivas proporcionadas al autor para crear y modificar sus presentaciones en el Sistema de Presentación [García, 01c]: **oculta asociación funcional** (Oculta-AF), **muestra asociación funcional** (Muestra-AF), **oculta asociación conceptual** (Oculta-AC) y **muestra asociación conceptual** (Muestra-AC).

La última acción evolutiva, *Muestra-AC, no necesita ser propagada* al Sistema de Aprendizaje, ya que ni las reglas de conocimiento ni las reglas de actualización requieren cambio alguno cuando se muestra una nueva asociación entre dos conceptos previamente existentes en la presentación sobre la que se definen.

Al contrario, las *tres primeras acciones sí requieren un proceso de propagación externa* sobre el Sistema de Aprendizaje. De modo que al ser ejecutadas en una determinada estructura conceptual de presentación, EC_P^k, se realizan de forma automática los cambios necesarios para garantizar la consistencia de los elementos definidos sobre ella en el Sistema de Aprendizaje.

Pueden existir varias estructuras de aprendizaje EC_A^1, EC_A^2,..., EC_A^n definidas a partir de la misma presentación EC_P^k. Por lo tanto, el proceso de propagación externa desencadenado tras una modificación en EC_P^k debe realizarse en todas y cada una de ellas, esto es $\forall\ EC_A^j$, con j = 1. .n.

A continuación se concreta el proceso de propagación externa acontecido tras cada una de las tres acciones evolutivas del Sistema de Aprendizaje indicadas más arriba. Además, en cada caso se hace una breve descripción de la utilidad de la acción para entender mejor la propagación desencadenada.

2.1 Muestra-AF(<c_o, r_f, i_j>, EC_P^k)

Esta acción evolutiva muestra en la presentación EC_P^k la relación r_f existente en la estructura conceptual de memorización entre el ítem $\mathbf{i_j}$ y el concepto c_o.

El concepto c_o debe haber sido incluido de antemano en la presentación. Pero, si el ítem i_j no está incluido previamente en la EC_P^k, esta acción evolutiva implica que se muestre también dicho ítem (propagación interna). Este hecho genera como consecuencia una propagación externa al Sistema de Aprendizaje que consiste en la creación por defecto de la regla de actualización para el nuevo ítem i_j a través de la ejecución de la acción evolutiva **CrearRu(i_j)** en el conjunto de estructuras de aprendizaje EC_A^1, EC_A^2,..., EC_A^n definidas a partir de EC_P^k.

2.2 Oculta-AF(<c_o, r_f, i_j>, EC_P^k)

Esta acción evolutiva oculta en la presentación indicada, la relación funcional r_f existente entre el ítem $\mathbf{i_j}$ y el concepto c_o.

Si el ítem i_j no está ligado a ningún otro concepto de la EC_P^k esta acción evolutiva implica un proceso de propagación interna que oculta, además de la relación funcional, el propio ítem, i_j. Cuando se omite un ítem en una presentación se hace necesario un proceso de propagación externa al Sistema de Aprendizaje. Esta propagación afecta al conjunto de reglas de actualización y de conocimiento definido sobre la EC_P^k.

Respecto al primero, Ru, se elimina la regla de actualización asociada a i_j usando la acción evolutiva **EliminarRu(i_j)**. También es necesario evaluar el cuerpo del resto de reglas de actualización y quitar, en caso de que existiera, el predicado de actualización que afecta al ítem ocultado. Así, toda regla Ru(i_g) que actualice el ítem i_j es modificada, eliminando tal predicado de actualización definido sobre un ítem ahora inexistente en EC_P^k. Para ello se ejecuta la acción evolutiva **EliminarPredicadoRu(i_g, i_j)**.

Del mismo modo, el conjunto de reglas de conocimiento, Rk, es modificado al eliminar de él, todas las reglas asociadas al ítem oculto. Para ello se ejecuta la acción evolutiva **EliminarRk(i_j, Rk-autor^t(i_j))** para t = 1. .n, siendo *n* el número total de reglas de conocimiento definidas por el autor para la visita a i_j. Asimismo, se eliminan del cuerpo del resto de reglas, las restricciones de conocimiento impuestas sobre i_j. Aplicando la acción evolutiva **EliminarPredicadoRk(i_g, Rk-autor^t(i_g), i_j)** para todas las reglas Rk-autor^t(i_g) que presenten en su cuerpo un predicado de conocimiento sobre i_j.

De nuevo, todas las propagaciones explicadas deben ejecutarse en todas las estructuras de aprendizaje EC_A^1, EC_A^2,..., EC_A^n definidas a partir de la EC_P^k modificada.

2.3 Oculta-AC(<c_o, r_c, c_d>, EC_P^k)

Esta acción evolutiva oculta la relación conceptual r_c que parte desde el concepto c_o y llega hasta el concepto c_d en la presentación EC_P^k.

Además, mediante propagación interna, se oculta todo aquel concepto que como consecuencia haya quedado desconectado en la presentación. Obviamente cuando se oculta un concepto c_k, también se ocultan todas las asociaciones funcionales r_f de éste, aplicando la acción evolutiva anterior, *Oculta-AF(<c_k, r_f, i_j>, EC_P^k)*, y su consecuente propagación al Sistema de Aprendizaje.

3. PROPAGACIÓN DESDE EL SISTEMA DE MEMORIZACIÓN

Son muchas y variadas, las acciones evolutivas disponibles en el Sistema de Memorización para construir y hacer evolucionar una estructura conceptual de memorización, de forma coherente con los cambios y ampliaciones surgidas en el dominio conceptual y de información del sistema hipermedia [García, 01c]. Básicamente este conjunto de acciones permite al autor crear, borrar y modificar los conceptos, ítems y asociaciones, tanto funcionales como conceptuales, que finalmente constituyen la estructura conceptual de memorización, EC_M.

A continuación se analiza el proceso de propagación para el subconjunto de acciones que una vez ejecutadas en el Sistema de Memorización requieren nuevos cambios en los elementos del Sistema de Aprendizaje. Algunos de estos cambios son propagados directamente desde el Sistema de Memorización al de Aprendizaje, mientras que otros lo hacen a través del Sistema de Presentación. Es decir, los cambios se propagan desde el Sistema de Memorización al de Presentación y como consecuencia al Sistema de Aprendizaje en la forma expuesta en la sección anterior.

Concretamente los cambios que tienen **propagación directa** desde el Sistema de Memorización hasta el Sistema de Aprendizaje son los que afectan al conjunto de *reglas de peso* y al *esquema del modelo de usuario*. Los cambios que se **propagan indirectamente** como consecuencia de los cambios propagados al Sistema de Presentación son los que afectan a las *reglas de conocimiento y de actualización*.

Para una misma estructura conceptual de memorización existen muy diversas estructuras de aprendizaje, cada una de las cuales se centra en una presentación concreta, y superpone a ésta distintas formas de recorrer la información ofrecida. Por este motivo:

a) De nuevo, los cambios en una EC_M que afectan a una presentación EC_P^k deben ser propagados al conjunto de reglas de actualización y conocimiento de todas las estructuras de aprendizaje EC_A^1, EC_A^2,..., EC_A^n definidas a partir de ella. Siendo el resultado de la propagación distinto para cada EC_A^j (con j = 1. .n) en función de las reglas de actualización y conocimiento que contiene.

b) Además, los cambios en una EC_M que afectan directamente al Sistema de Aprendizaje deben ser reflejados en todas las estructuras EC_A^1, EC_A^2,..., EC_A^m definidas a partir de ella. Pero en esta ocasión, el resultado de la propagación es el mismo para todas las estructuras de aprendizaje, ya que todas presentan el mismo

conjunto de *reglas de peso* y la misma *estructura del modelo de usuario*. Por lo que la propagación es ejecutada una sola vez y actualizada, con j = 1. . m, en todas las EC_A^j.

A continuación se describen brevemente las acciones evolutivas del Sistema de Memorización que requieren propagación sobre el Sistema de Aprendizaje, indicando en cada caso en qué consiste dicha propagación.

3.1 Crea-concepto(et, EC_M)

Esta acción evolutiva crea un nuevo concepto en la estructura conceptual de memorización EC_M con el nombre indicado, esto es **et**.

Esto obliga a que se añada una nueva entrada asociada al nuevo concepto en la estructura del modelo de usuario definido para esa EC_M. Esta entrada es de tipo "concepto" y por lo tanto, el identificador del concepto coincide con el nombre del mismo. De modo que la acción evolutiva del Sistema de Aprendizaje que se ejecuta para esta propagación es **AñadirElementoMU(et, "concepto", et)**.

Además, aunque el concepto no tenga aún ítems asociados, es necesario crear una regla de peso inicial para el nuevo concepto. Esta propagación se realiza mediante la acción evolutiva **CrearRw(et)**.

En ambos casos, la propagación realizada afecta al conjunto de estructuras de aprendizaje, EC_A^1, EC_A^2,..., EC_A^m, creadas a partir de la estructura conceptual de memorización modificada. Por lo tanto debe ser actualizada en todas ellas.

3.2 Crea-Ac-y-concepto(c_o, et_c, et_r, EC_M)

Esta acción evolutiva crea un nuevo concepto con el nombre et_c y una asociación conceptual etiquetada como et_r que une el concepto existente c_o con el concepto creado.

La creación de una nueva asociación conceptual no tiene efecto sobre el Sistema de Aprendizaje pero sí la creación de un nuevo concepto. Por lo tanto, la propagación realizada en este caso coincide con la explicada en el punto anterior, esto es *Crea-concepto(et_c, EC_M)*.

3.3 Crea-asociación-funcional(c_o, i_j, et, EC_M)

Esta acción evolutiva crea una relación funcional etiquetada con el nombre et entre el concepto $\mathbf{c_o}$ y el ítem $\mathbf{i_j}$, ambos existentes en la estructura conceptual de memorización EC_M.

Esto implica que en la regla de peso asociada al concepto c_o es necesario incluir un nuevo término asociado al ítem i_j recientemente asociado a él. Esta propagación es realizada automáticamente a través de la acción evolutiva **AñadirTérminoRw(c_o, i_j)** del Sistema de Aprendizaje. El resultado de la propagación es actualizado en todas las estructuras conceptuales de aprendizaje, EC_A^1, EC_A^2,..., EC_A^m, definidas sobre la EC_M modificada.

3.4 Borra-asociación-funcional(<c_o, r_f, i_j>, EC_M)

Esta acción evolutiva borra la relación funcional r_f que asocia en la estructura conceptual de memorización EC_M el ítem $\mathbf{i_j}$ al concepto $\mathbf{c_o}$. No hay que comprobar si tras eliminar r_f el ítem queda desconectado, ya que siempre está unido, al menos, con el concepto sistema.

Este cambio requiere modificar la regla de peso del concepto c_o, eliminando en ella el término asociado al ítem i_j que se acaba de desligar del concepto. Esta propagación se realiza automáticamente ejecutando la acción evolutiva del Sistema de Aprendizaje **EliminarTérminoRw(c_o, i_j)**. El resultado de la propagación es actualizado en las *m* estructuras de aprendizaje creadas a partir de EC_M, esto es en EC_A^1, EC_A^2,..., EC_A^m.

Además, la relación funcional eliminada en EC_M debe ser ocultada en todas las presentaciones creadas a partir de ésta. Por lo tanto, suponiendo *p* estructuras conceptuales de presentación diferentes, para k = 1. .p se ejecuta la acción evolutiva del Sistema de Presentación *Oculta-AF (<c_o, r_f, i_j>, EC_P^k)*. Lo cual genera la propagación al Sistema de Aprendizaje explicada en la sección 2.

3.5 Modifica-asociación-funcional(<c_o, r_f, i_j>, o_n, EC_M)

Esta acción evolutiva modifica el concepto origen o el ítem destino de una asociación funcional existente en la EC_M. Así pues, o_n representa el nuevo concepto origen o el nuevo ítem destino de la asociación según cuál sea el cambio aguardado por el autor. Esto es, $o_n \in I(EC_M) \vee o_n \in C(EC_M)$.

En cualquier caso, la ejecución de la acción evolutiva se descompone en dos operaciones:

Primero se borra la asociación funcional actualmente existente entre el concepto c_o y el ítem i_j. Se ejecuta para ello la acción evolutiva *Borra-asociación-funcional(<c_o, r_f, i_j>, EC_M)*, propagando los cambios necesarios sobre el Sistema de Aprendizaje, tal y como se explicó en la subsección anterior.

Después:

a) Si $o_n \in I(EC_M)$ se crea una nueva asociación funcional del concepto c_o con el ítem o_n. Se ejecuta la acción evolutiva *Crea-asociación-funcional((<c_o, r_f, o_n>, EC_M)* con su consecuente propagación sobre el Sistema de Aprendizaje (subsección 3.3).

b) Si $o_n \in C(EC_M)$ se crea una nueva asociación funcional del concepto o_n con el ítem i_j. Se ejecuta y propaga la acción evolutiva *Crea-asociación-funcional((<o_n, r_f, i_j>, EC_M)*.

3.6 Borra-asociación-conceptual(<c_o, r_c, c_d>, EC_M)

Esta acción evolutiva borra la relación conceptual r_c existente en la EC_M entre los conceptos c_o y c_d.

Por propagación externa, la relación eliminada en EC_M debe ser ocultada en todas las presentaciones EC_P^k que la contengan. Esta propagación al Sistema de Presentación se

realiza mediante la acción evolutiva *Oculta-AC(<c_o, r_c, c_d>, EC_P^k)*, que como se explicó en la subsección 2.3 también repercute sobre el Sistema de Aprendizaje.

Asimismo, como consecuencia de la modificación y por propagación interna, se eliminan en la EC_M todos los conceptos que como consecuencia del cambio han quedado aislados. Este hecho requiere un proceso de propagación directa sobre el modelo de usuario y las reglas de peso definidas para la EC_M en todas las estructuras de aprendizaje creadas sobre ésta. Concretamente, cuando se elimina un concepto $\mathbf{c_k}$ en EC_M:

- Es necesario eliminar su entrada en el modelo de usuario. Esto se realiza mediante la ejecución de la acción evolutiva del Sistema de Aprendizaje **EliminarElementoMU(c_k , "concepto")**.

- También hay que eliminar la regla de peso $Rw(c_k)$ asociada al concepto que deja de existir. Esta propagación se realiza mediante la acción **EliminarRw(c_k)**.

Por último, cuando se elimina un concepto c_k también es obligado eliminar todas sus asociaciones funcionales, lo que implica la ejecución automática de la acción evolutiva *Borra-asociación-funcional(<c_k, r_f, i_j>, EC_M)* para cada ítem i_j asociado al concepto eliminado. Esta propagación interna dentro del Sistema de Memorización genera un nuevo proceso de propagación externa sobre el Sistema de Aprendizaje (explicado en la subsección 3.4).

3.7 Modifica-asociación-conceptual(c_n, <c_o, r_c, c_d>, EC_M)

Esta acción evolutiva permite cambiar el concepto destino u origen de una relación conceptual por c_n.

Como consecuencia será necesario eliminar los conceptos c_k desconectados tras la modificación, así como las asociaciones conceptuales y funcionales implicadas. Por lo tanto, el proceso de propagación es idéntico al explicado en la subsección anterior.

3.8 Cambia-etiqueta(c_k, et', EC_M)

Esta acción evolutiva puede ser utilizada por el autor para modificar el nombre de un concepto $\mathbf{c_k}$ de la estructura conceptual de memorización EC_M, asignándole la nueva etiqueta **et'**.

El cambio de nombre debe ser actualizado en la entrada correspondiente al concepto en el modelo de usuario, pero también en su regla de peso ya que el nombre actúa como identificador del concepto. Las acciones evolutivas del Sistema de Aprendizaje aplicadas para ello son respectivamente: **ModificarElementoMU(c_k, "concepto", et')** y **CambiarNombreRw(c_k, et')**. Por supuesto la propagación realizada debe ser actualizada en todas las estructuras de aprendizaje, EC_A^1, EC_A^2,..., EC_A^m, definidas sobre la EC_M modificada.

3.9 Crea-ítem(EC_M)

Esta acción evolutiva permite al autor introducir un nuevo ítem de información en el sistema hipermedia. Además del contenido, el autor establece una serie de propiedades para el ítem como son: **nombre**, dificultad, autor, fecha de creación, idioma, medio, etc. El sistema se encarga de generar y asignar automáticamente al ítem un identificador $\mathbf{i_j}$ que permitirá distinguirlo unívocamente del resto. El ítem creado es asociado por defecto al concepto *Sistema* mediante una relación genérica. Después el autor podrá ligarlo a cuantos conceptos desee a través de la acción evolutiva Crea-asociación-funcional.

La única propagación externa al Sistema de Aprendizaje se realiza de forma directa y consiste en incluir una entrada en el modelo de usuario asociada al ítem i_j que se acaba de crear. Para ello se ejecuta la acción evolutiva **AñadirElementoMU(i_j, "ítem", nombre)**. De nuevo se actualiza el resultado en todas las estructuras de aprendizaje donde sea necesario.

3.10 Borra-ítem(i_j, EC_M)

Esta acción evolutiva es usada para eliminar un ítem $\mathbf{i_j}$ de la estructura conceptual de memorización EC_M.

Como consecuencia es necesario borrar la entrada asociada al ítem en el modelo de usuario, empleando para ello la acción evolutiva del Sistema de Aprendizaje **EliminarElementoMU(i_j, "ítem")**.

Además de esta modificación directa, existen otros cambios sobre el Sistema de Aprendizaje derivados de la necesidad de suprimir, también, todas las asociaciones funcionales que tienen como destino el ítem eliminado. Esto es, de ejecutar por propagación interna la acción evolutiva del Sistema de Memorización *Borra-asociación-funcional*($<c_o, r_f, i_j>$, EC_M) para todas las asociaciones r_f que tiene el ítem i_j en EC_M. Tal y como se explicó en la subsección 3.4, esta acción genera cambios en el Sistema de Presentación que repercuten nuevamente sobre el Sistema de Aprendizaje.

4. RECOPILACIÓN DE CAMBIOS QUE AFECTAN AL SISTEMA DE APRENDIZAJE

En las secciones 2 y 3 del presente capítulo se han estudiado aquellas acciones evolutivas ofrecidas por los Sistemas de Memorización y Presentación, cuya ejecución plantea la necesidad de evaluar y emprender un proceso de propagación externa sobre el Sistema de Aprendizaje. En este apartado se identifican y resumen aquellos cambios que, independientemente de la acción evolutiva que los produce, exigen nuevas modificaciones sobre algunos elementos del Sistema de Aprendizaje.

En la primera columna de la tabla 1 se muestran los nueve cambios que al realizarse sobre uno de estos sistemas pueden implicar un proceso de propagación sobre uno o varios elementos del Sistema de Aprendizaje. En la segunda columna de la tabla, se especifica para cada cambio si es realizado sobre una EC_M o una EC_P. En la tercera columna se explica la propagación realizada en el Sistema de Aprendizaje, y en la última columna se indica concretamente los elementos que se modifican.

En dicha tabla se pone de manifiesto cómo los cambios en una estructura conceptual de memorización afectan directamente al modelo de usuario y a las reglas de peso definidas para ésta en el Sistema de Aprendizaje. Mientras que los cambios realizados sobre el Sistema de Presentación tienen repercusión sobre el conjunto de reglas de conocimiento y actualización establecido sobre la estructura conceptual de presentación modificada. Sin olvidar que, tal y como se explicó en secciones anteriores, determinados cambios en el Sistema de Memorización son propagados al Sistema de Presentación y como consecuencia indirecta al Sistema de Aprendizaje.

Tabla 1: Cambios propagados sobre el Sistema de Aprendizaje

	Cambio	en	Propagación	sobre
A	Crear concepto	EC_M	Implica añadir una nueva entrada en el modelo de usuario asociada al concepto creado. También requiere generar la regla de peso inicial para el nuevo concepto.	MU y Rw
B	Borrar concepto		Implica eliminar la entrada del modelo de usuario asociada al concepto desaparecido. Asimismo, es necesario eliminar la regla de peso del concepto borrado.	
C	Renombrar concepto		Implica modificar el nombre del concepto en la entrada del modelo de usuario. Y en la regla de peso asociada al concepto renombrado.	MU y Rw
D	Crear ítem		Supone la adición de una nueva entrada en el modelo de usuario asociada al ítem creado.	MU
E	Borrar ítem		Requiere eliminar la entrada del ítem borrado en el modelo de usuario.	
F	Crear AF		Implica añadir un nuevo término en la regla de peso del concepto implicado.	Rw
G	Borrar AF		Requiere eliminar un término en la regla de peso del concepto implicado.	
H	Mostrar ítem	EC_P	Supone generar la regla de actualización por defecto para el nuevo ítem mostrado en la presentación.	Ru
I	Ocultar ítem		Implica la eliminación de la regla de actualización asociada al ítem ocultado y de todas sus reglas de conocimiento. También es necesario eliminar la ocurrencia del ítem en el cuerpo del resto de reglas, tanto de actualización como de conocimiento.	Ru y Rk

La tabla 2 marca para cada cambio de la tabla 1, la acción o acciones evolutivas del Sistema de Aprendizaje que son ejecutadas, de forma automática por el sistema, durante el proceso de propagación externa. Por cuestión de espacio, se utiliza la letra (A-I) asociada a cada cambio en la tabla 1 en lugar del nombre completo del cambio.

La interpretación de la tabla puede hacerse por columnas o por filas.

Por ejemplo, si estudiamos la fila correspondiente a la acción evolutiva EliminarElementoMU vemos que ésta se ejecuta como consecuencia de dos cambios

Borrar concepto (B) y Borrar ítem (E). Si estudiamos la columna asociada al cambio Crear concepto (A) vemos que desencadena la ejecución de dos acciones evolutivas: AñadirElementoMU y CrearRw.

Tabla 2: ACe del Sistema de Aprendizaje ejecutadas para cada cambio

A	B	C	D	E	F	G	H	I	ACe
√			√						**AñadirElementoMU**
	√			√					**EliminarElementoMU**
		√							**ModificarElementoMU**
√									**CrearRw**
	√								**EliminarRw**
					√				**AñadirTérminoRw**
						√			**EliminarTérminioRw**
		√							**ModificarNombreRw**
							√		**CrearRu**
								√	**EliminarRu**
								√	**EliminarPredicadoRu**
								√	**EliminarRk**
								√	**EliminarPredicadoRk**

Por último, la tabla 3 relaciona los cambios considerados en la tabla 1 con las acciones evolutivas del Sistema de Presentación y de Memorización que requieren ser propagadas al Sistema de Aprendizaje (secciones 2 y 3). Para cada acción evolutiva se marca con el símbolo √ los cambios que ésta siempre produce al ser ejecutada, y con el símbolo × aquellos que a veces produce y otras no, dependiendo del estado actual de la estructura conceptual sobre la que se ejecuta.

Por ejemplo, si nos fijamos en la acción evolutiva Borra-ítem (última fila de la tabla) vemos que evidentemente siempre ocasiona el cambio E (Borrar ítem). Asimismo siempre implica el cambio G (Borrar AF) para cada una de sus asociaciones funcionales (al menos tiene una con el concepto Sistema). También puede generar el cambio I (Ocultar ítem). Pero esto sólo ocurre si existe alguna presentación que muestre el ítem eliminado, por este motivo dicho cambio es marcado con el símbolo ×.

La acción evolutiva Borra-asociación-funcional siempre ocasiona el cambio G (Borra AF), y en ocasiones puede dar lugar al cambio I (Ocultar ítem). Este último cambio sólo tiene lugar si al eliminar la relación funcional, el ítem implicado ha quedado desconectado en la presentación. Si dicho ítem tenía otras asociaciones funcionales en la presentación no se oculta. Por este motivo nuevamente el cambio I es marcado con ×.

De esta forma queda perfectamente establecido el proceso de propagación externa que permite mantener coherente el Sistema de Aprendizaje en relación a los cambios surgidos en el resto de sistemas del modelo SEM-HP.

Tabla 3: Cambios generados al ejecutar una acción evolutiva

ACe \ Cambio		A	B	C	D	E	F	G	H	I
Sistema de Presentación	**Muestra-AF**								×	
	Oculta-AF									×
	Oculta-AC									×
Sistema de Memorización	**Crea-concepto**	√								
	Crea-Ac-y-concepto	√								
	Crea-asociación-funcional						√			
	Borra-asociación-funcional							√		×
	Modifica-asociación-funcional						√	√		×
	Borra-asociación-conceptual		×					×		×
	Modifica-asociación-conceptual		×					×		×
	Cambia-etiqueta			√						
	Crea-ítem				√					
	Borra-ítem					√		√		×

5. SECUENCIAS DE PROPAGACIÓN DEL CAMBIO

Una vez estudiada la necesidad de propagar algunos de los cambios que se realizan en los Sistemas de Memorización y Presentación hasta el Sistema de Aprendizaje, vamos a describir con más exactitud en qué consiste el proceso de propagación externa del cambio. Para ello, separamos y definimos las dos secuencias de propagación posibles: desde el Sistema de Presentación y desde el Sistema de Memorización.

> **Def 13.1 [Propagación externa del cambio]** El mecanismo de propagación externa del cambio es el que se ocupa de garantizar que todos los sistemas co-evolucionen en una forma consistente después de que el autor realice una modificación en cualquiera de ellos. De esta forma además de la consistencia individual de cada sistema se garantiza la consistencia global del conjunto, asegurando pues, la integridad del sistema hipermedia en todo momento.

Para ilustrar el proceso de propagación externa que tiene lugar sobre el Sistema de Aprendizaje nos ayudaremos del diagrama de clases parcial de SEM-HP mostrado en la figura 1. En ella aparecen de izquierda a derecha y de arriba a abajo las siguientes clases: Interfaz, SEM-HP, Sistema, Sistema de memorización, Estructura conceptual de memorización (ECm), Diccionario de Datos (DD), Sistema de presentación, Estructura conceptual de presentación (ECp), Sistema de navegación, Sistema de aprendizaje, Reglas de peso (Rw), Modelo de usuario (MU), Reglas de actualización (Ru), Reglas de conocimiento (Rk), Meta-sistema, Acciones evolutivas (ACe), Restricciones (RT), Evaluador y Propagador.

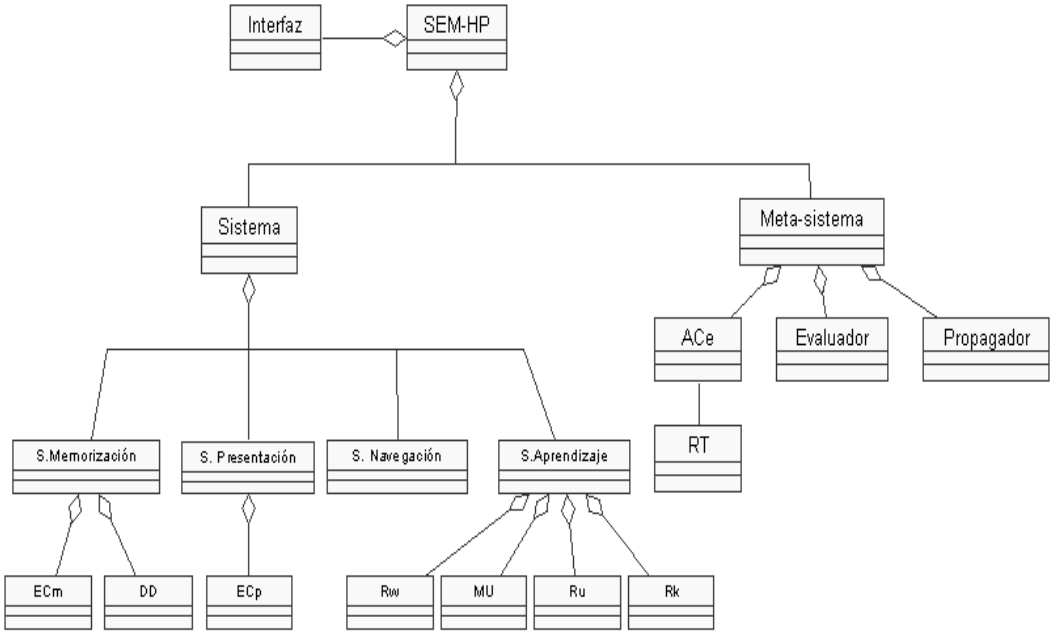


Figura 1: Diagrama de clases parcial de SEM-HP

Todos los elementos incluidos en la figura han sido previamente definidos excepto los módulos **Evaluador** y **Propagador**. Ambos módulos son parte del Metasistema del modelo SEM-HP y su utilidad es respectivamente: comprobar la satisfacción de las restricciones (pre o pos) impuestas para la ejecución de una acción evolutiva y determinar si es necesario un proceso de propagación externa tras el cambio y en caso afirmativo qué acciones evolutivas se ejecutan durante esa propagación.

5.1 Secuencia de propagación desde el Sistema de Presentación

La secuencia de propagación sobre el Sistema de Aprendizaje que tiene lugar desde el Sistema de Presentación puede explicarse de la siguiente forma:

Supongamos que el autor a través de la correspondiente *Interfaz* solicita realizar una modificación sobre algún elemento del Sistema de Presentación, por ejemplo mostrar una nueva asociación funcional. En este caso *ACe* = Muestra-AF.

El módulo *Evaluador* del Metasistema se encargaría de verificar si la ejecución de esa ACe, sobre la correspondiente estructura conceptual de presentación, ECp, satisface el conjunto de restricciones asociadas, RTs. En este caso, las RTs exigen que el concepto implicado en la nueva asociación funcional exista previamente en la ECp.

En caso positivo, el *Evaluador* ejecuta la acción requerida y la propagación interna necesaria. En este caso, la propagación interna muestra en la ECp el ítem implicado en la nueva asociación funcional si éste actualmente no se muestra.

Después el *Evaluador* solicita al *Propagador* la lista de acciones evolutivas, listaACe, que como extensión de la actual deben ser realizadas sobre el Sistema de Aprendizaje. En este caso, la propagación externa consiste en crear, si no existe previamente, la regla de actualización por defecto para el ítem implicado en la asociación, pero sólo, en las estructuras de aprendizaje definidas sobre la ECp modificada.

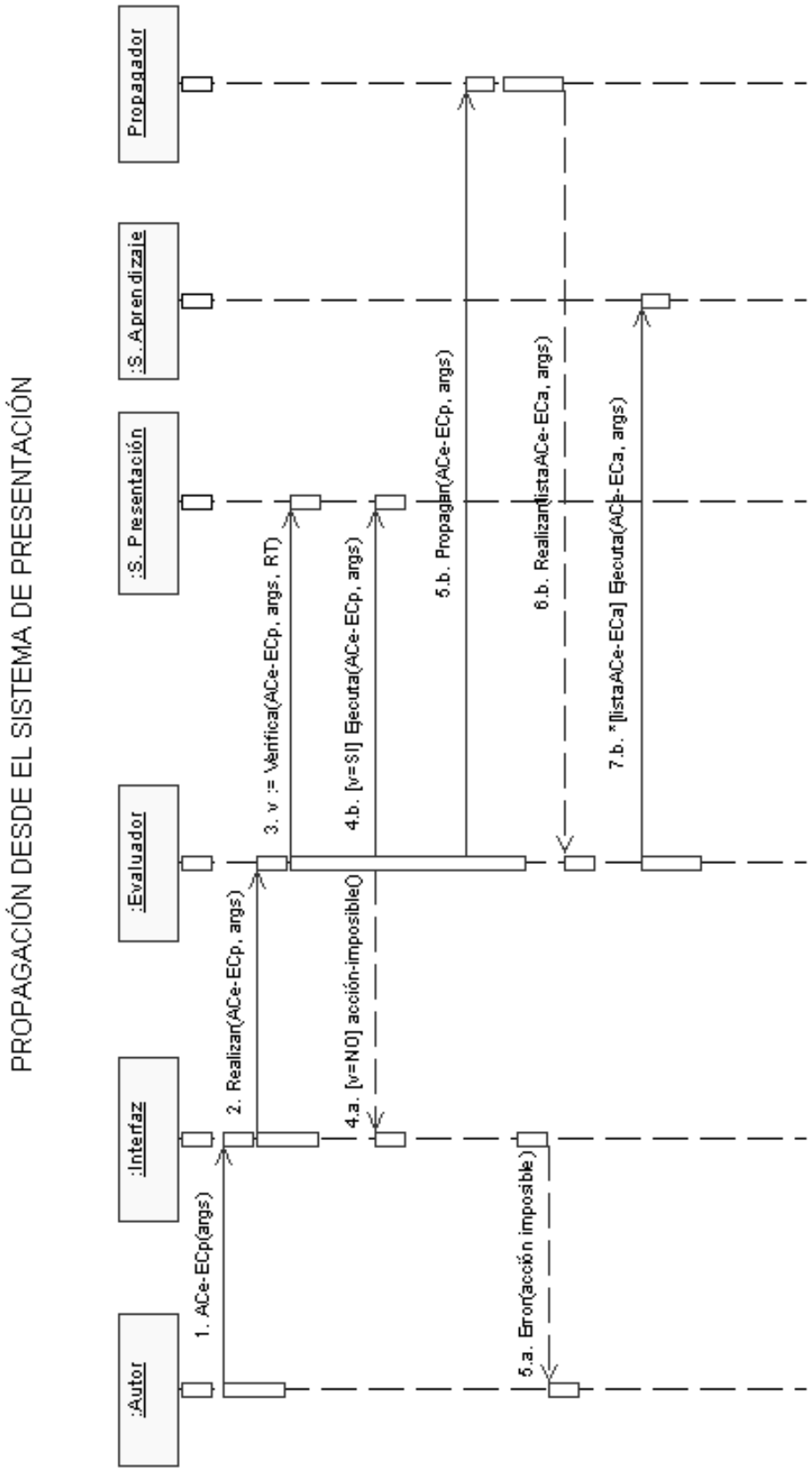


Figura 2: Secuencia de propagación desde el S. Presentación

En la figura 2 se muestra gráficamente y de modo general la secuencia de propagación del cambio explicada. Se han utilizado ACe-ECp y ACe-ECa para hacer referencia a una acción evolutiva del Sistema de Presentación y del Sistema de Aprendizaje respectivamente.

Después de que el Evaluador ha verificado las restricciones de la acción solicitada por el autor (1, 2 y 3), se abren dos secuencias distintas, según la acción no deba ejecutarse (4.a, 5.a) o sí (4.b, 5.b, 6.b, 7.b). En el primer caso simplemente se informa al autor de la imposibilidad de realizar la acción. Mientras que en el segundo se ejecuta la acción evolutiva sobre el Sistema de Presentación y se solicita al Propagador la lista de acciones que deben ser ejecutadas sobre el Sistema de Aprendizaje para propagar dicho cambio.

De forma resumida, la secuencia que tiene lugar cuando el autor solicita la ejecución de una acción evolutiva en el Sistema de Presentación, es la siguiente:

Paso 1. El Evaluador comprueba las restricciones de la acción evolutiva solicitada. Si no se satisfacen todas ellas, informa al autor de la imposibilidad de ejecutar la acción (Error) y Termina. En otro caso sigue al Paso 2.

Paso 2. El Evaluador ejecuta la acción evolutiva sobre la estructura conceptual de presentación. Si es necesario, realiza un proceso de propagación interna sobre dicha estructura.

Paso 3. El Evaluador solicita al Propagador la secuencia de acciones evolutivas que debe realizar sobre el Sistema de Aprendizaje.

Paso 4. El Propagador devuelve al Evaluador el conjunto de acciones de propagación externa. Si dicha secuencia es vacía Termina. En otro caso sigue al Paso 5.

Paso 5. El Evaluador ejecuta sobre el Sistema de Aprendizaje las acciones evolutivas indicadas por el Propagador. Concretamente, estas acciones evolutivas afectan a las reglas de conocimiento y/o actualización. Y sólo son ejecutadas sobre las estructuras conceptuales de aprendizaje creadas a partir de la presentación modificada.

Termina.

Algoritmo 1. Secuencia de propagación desde el Sistema de Presentación

5.2 Secuencia de propagación desde el Sistema de Memorización

La secuencia de propagación que tiene lugar tras los cambios realizados en el Sistema de Memorización es similar a la explicada para el Sistema de Presentación, salvo que en este caso, pueden producirse cambios en el Sistema de Aprendizaje tanto directa como indirectamente.

Los cambios que se producen **directamente** (D) en el Sistema de Aprendizaje como propagación de un cambio realizado sobre la estructura conceptual de memorización afectan a la estructura del *modelo de usuario* y al conjunto de *reglas de peso*. Estos cambios consisten básicamente en:

- Añadir y eliminar ítems y conceptos en la EC_M.
- Modificar las asociaciones funcionales existentes la EC_M.

Los cambios que se producen **indirectamente** (I) ocurren como consecuencia de la propagación de los cambios realizados en el Sistema de Memorización al Sistema de Presentación y después al Sistema de Aprendizaje. Estos cambios afectan por tanto, a las *reglas de actualización y conocimiento* y son aquellos que implican eliminar un ítem en la estructura conceptual de memorización y ocultarlo en todas las presentaciones donde se muestra.

La secuencia de propagación desde el Sistema de Memorización al Sistema de Aprendizaje se resume en los siguientes pasos:

Paso 1. El Evaluador comprueba las restricciones de la acción evolutiva solicitada. Si éstas se satisfacen, ejecuta la acción evolutiva sobre la estructura conceptual de memorización y la propagación interna que proceda. En caso de no satisfacerse da Error y Termina.

Paso 2. El Evaluador solicita al Propagador la secuencia de acciones evolutivas que debe ejecutar, como consecuencia directa de la modificación realizada en la estructura conceptual de memorización. Esta propagación es realizada sobre el Sistema de Presentación y sobre el Sistema de Aprendizaje. Siendo estos últimos cambios de tipo **D**.

Paso 3. Después de ejecutar los cambios del paso anterior, el Evaluador solicita al Propagador el conjunto de cambios de tipo **I** que deben ejecutarse en el Sistema de Aprendizaje como consecuencia de los cambios realizados sobre el Sistema de Presentación. Una vez ejecutados Termina.

Termina.

Algoritmo 2. Secuencia de propagación desde el Sistema de Memorización

En la figura 3 se muestra gráficamente la secuencia de propagación del cambio explicada en el algoritmo 2. Se han utilizado ACe-ECm, ACe-ECp y ACe-ECa para hacer referencia a una acción evolutiva del Sistema de Memorización, del Sistema de Presentación y del Sistema de Aprendizaje respectivamente.

Una vez que el Evaluador ha verificado las restricciones de la acción solicitada por el autor (1, 2 y 3), se abren nuevamente dos secuencias distintas, según la acción no deba ser ejecutada (4.a, 5.a) o sí. En este último caso, el Evaluador ejecuta la acción evolutiva sobre el Sistema de Memorización (4.b) y solicita al Propagador la lista de acciones que debe ejecutar para propagar el cambio (5.b).

La respuesta del Propagador es dividida en dos: la propagación sobre el Sistema de Aprendizaje (6.b.1) y la propagación sobre el Sistema de Presentación (6.b.2). El Evaluador ejecuta las acciones evolutivas correspondientes a la propagación directa sobre el Sistema de Aprendizaje (7.b.1) y sobre el Sistema de Presentación (7.b.2).

Después (8.b.2), el Evaluador solicita al Propagador las acciones evolutivas que debe ejecutar sobre el Sistema de Aprendizaje como consecuencia indirecta de la propagación realizada sobre el Sistema de Presentación. Y, una vez obtenida la respuesta (9.b.2), ejecuta las acciones que se le indican (10.b.2).

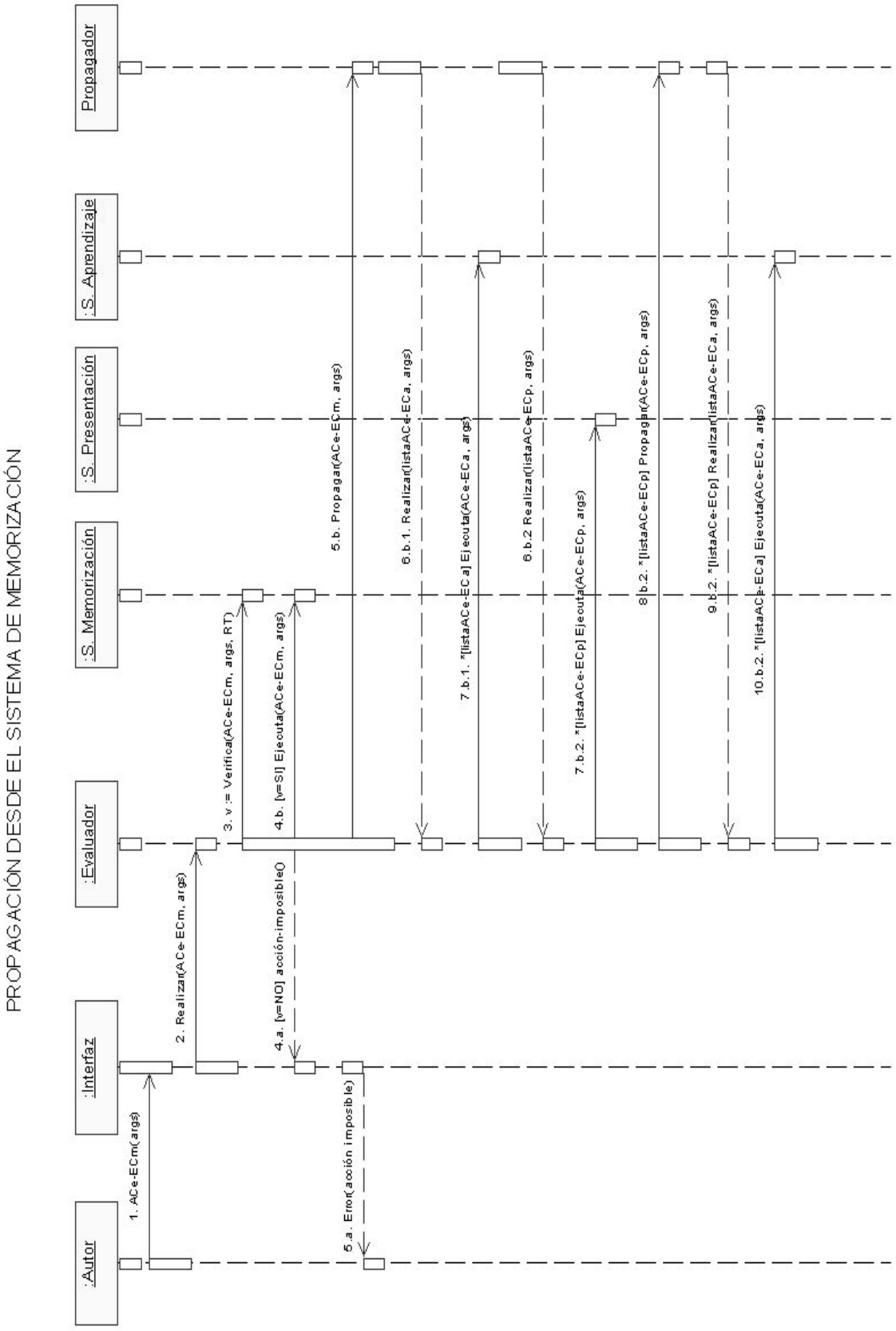
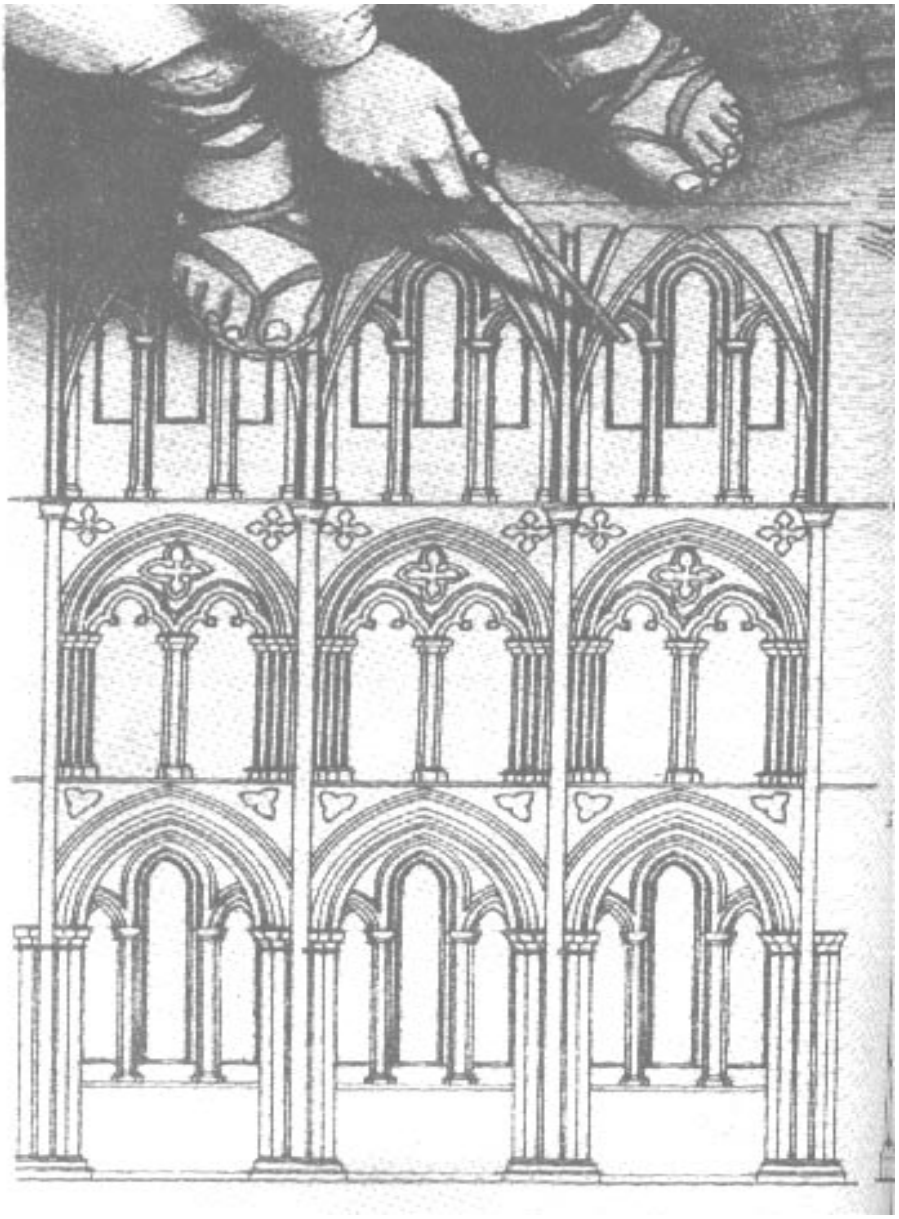


Figura 3: Secuencia de propagación desde el S. Memorización

Módulo IV

Sistema de Aprendizaje: Elementos Funcionales

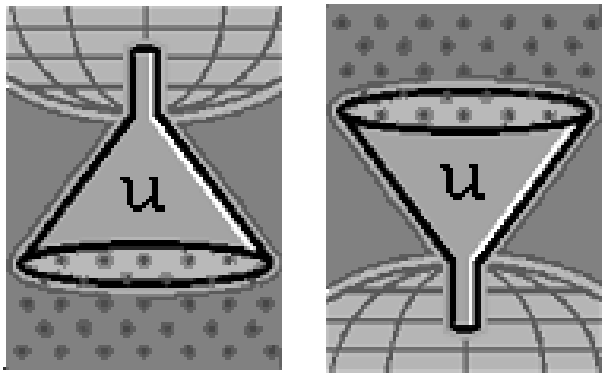


Los Pilares de la Tierra

Módulo IV

CAPÍTULO 14

Elección de la EC_A



Resumen

En el presente capítulo se describe la capacidad del sistema para elegir de forma automática la estructura conceptual de aprendizaje que mejor se ajusta al perfil de cada usuario. Para llevar a cabo la elección, previamente el autor debe etiquetar cada estructura con el subdominio o subdominios de conocimiento que captura, y la experiencia, en la materia y de navegación, que deben tener los usuarios para poder hacer un uso adecuado de ésta. Se justifican las características incluidas en el etiquetado y se facilitan criterios generales para asignarles valor. También se proporciona un conjunto de acciones evolutivas que permiten hacer evolucionar el etiquetado de una estructura de acuerdo a los cambios acontecidos en la misma. Finalmente, se detalla el procedimiento seguido por el sistema para elegir la mejor estructura de entre todas las posibles.

Tabla de contenidos

Elección de la EC_A

1. INTRODUCCIÓN

Como se describió, en el capítulo 6 y se muestra en la figura 1, el modelo SEM-HP permite crear a partir de una estructura conceptual de memorización, EC_M, otras estructuras más locales y abarcables, denominadas estructuras conceptuales de presentación, EC_P^1, EC_P^2,..., EC_P^n. Después, sobre cada presentación, el autor define uno o varios conjuntos de reglas de orden y navegabilidad, RTnb ∪ Ro, que convierten las relaciones conceptuales existentes en enlaces de navegación. De este modo, para una misma presentación, el autor construye distintas estructuras conceptuales de navegación, EC_N^1, EC_N^2,..., EC_N^r. De nuevo, siguiendo este **enfoque incremental**, el autor puede definir diferentes conjuntos de reglas de aprendizaje, Rk ∪ Ru, sobre una misma estructura conceptual de navegación, dando lugar finalmente, a un amplio abanico de estructuras conceptuales de aprendizaje, EC_A^1, EC_A^2,..., EC_A^m.

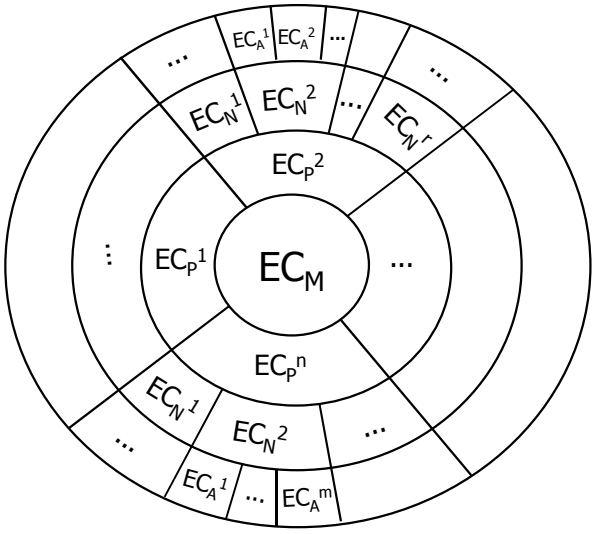


Figura 1: Enfoque incremental para definir una EC_A

Todos los usuarios de un sistema hipermedia tienen en común la estructura conceptual de memorización central que representa el dominio de conocimiento completo. Sin embargo, para que el usuario pueda recorrer la información del sistema, es necesario proporcionarle una estructura conceptual de aprendizaje que defina una presentación, navegación y aprendizaje concretos. La cuestión es ¿cuál de entre todas las EC_A posibles?

La respuesta a esta pregunta se expone y razona ampliamente en el presente capítulo. Sin embargo, en una primera aproximación parece obvio que sea cuál sea la EC_A elegida, dicha elección debe hacerse de forma personalizada para cada usuario. Lo cuál plantea otra cuestión: ¿Qué características del usuario deben tenerse en cuenta para dicha elección? De nuevo sin concretar la respuesta, es posible afirmar que independientemente de las características consideradas, cuando éstas cambien sensiblemente también debería cambiar la estructura elegida para el usuario.

De este modo, la EC_A elegida para un usuario no se asocia de manera permanente a éste, es decir, no se le proporciona la misma EC_A en todas sus sesiones de trabajo dentro del

sistema. La evolución que experimente el usuario, por ejemplo como consecuencia de su proceso de navegación, es tenida en cuenta para, en cada momento, proporcionarle la EC_A que mejor se ajuste a él en su situación actual. Por este motivo, como veremos más adelante, la EC_A que un usuario utiliza depende de éste y evoluciona con él, entre sesiones e incluso durante una misma sesión si así se requiere.

Suponiendo ya fijadas las características del usuario que se van a tener en cuenta para elegir su EC_A, cabe proponer una última cuestión importante, ¿cómo se sabe si una EC_A es adecuada o no para un usuario? Seguramente si el autor del sistema y ese usuario pudiesen estar en contacto, el autor preguntaría al usuario por esas características y revisando las EC_A que ha definido no tardaría demasiado en ofrecerle una.

El procedimiento mental que realiza el autor es sencillo, simplemente compara las características del usuario con las de cada EC_A, optando por aquella que considera más próxima a éstas. El autor puede hacer esa comparación porque posee cierto conocimiento sobre las estructuras que ha creado. Pero, ¿y el sistema?, ¿cómo puede hacer dicha elección de forma automática?

El sistema posee mucha información del usuario en el modelo que internamente mantiene sobre éste (capítulo 12, Modelo de usuario). Por lo tanto, para conocer las características que necesita del usuario sólo tiene que consultarlas en su modelo. Sin embargo, para poder cotejar esa información con cada EC_A debe tener también alguna información acerca de éstas. Dicha información puede ser proporcionada por el autor cuando crea las estructuras, de forma que al final, cada EC_A esté etiquetada con las características requeridas por el sistema para hacer la citada comparación.

2. ETIQUETADO DE UNA EC_A

Las características que el autor debe indicar en cada EC_A son las mismas que después el sistema tiene en cuenta para determinar qué estructura proporciona a cada usuario.

Def 14.1 [Etiquetado de EC_A] El etiquetado de una estructura conceptual de aprendizaje, EC_A, es un conjunto de *n* parejas ($atributo^i$, $etiqueta^i$), donde, desde i = 1. .n, la $etiqueta^i$ refleja el valor que dicha estructura tiene para la característica $atributo^i$. Esta información se usa para elegir la EC_A que se proporciona a un usuario de acuerdo a su perfil.

$$etiquetado(EC_A) = \{(atributo^1, etiqueta^1),..., (atributo^n, etiqueta^n)\} \quad (1)$$

2.1 Características utilizadas en el etiquetado

Puesto que es el autor quien debe proporcionar el valor de cada atributo del etiquetado, el número de características consideradas no debe ser excesivo para evitarle un esfuerzo adicional. Pero sí suficiente para recoger el perfil del usuario y en función de éste poder ofrecerle la EC_A que mejor se le ajuste.

Las características del usuario consideradas para elegir su EC_A no deben ser demasiado puntuales, como por ejemplo, su grado de conocimiento sobre un concepto, el conocimiento mínimo que desea alcanzar sobre un determinado ítem, sus gustos respecto al idioma, ni mucho menos su nombre, apellidos, etc.

Por el contrario, interesan características más generales que permitan agrupar a los usuarios en perfiles o grupos, donde dos usuarios con el mismo perfil, a pesar de sus múltiples diferencias puntuales, tengan una significativa base común. De esta forma, a todos los usuarios con un mismo perfil, el sistema les proporciona la misma EC_A para trabajar. Y después, aplicando diversas técnicas de adaptación, dicha estructura se acomoda completamente a las diferencias específicas de cada uno.

De esta forma, aunque todos los usuarios con idéntico perfil recorran la misma presentación y compartan las mismas reglas de navegación y adquisición de conocimiento, la adaptación de la estructura proporcionada será distinta en función de cuál sea su estado de conocimiento concreto, el subconjunto de ítems que le interesan o sus gustos respecto al contenido de los ítems. La adaptación de la EC_A elegida se explica en capítulos posteriores (capítulos del 16 al 21).

En esta sección nos centramos en describir cuáles son esas características más genéricas que definen el perfil del usuario y por tanto, deben ser consideradas para elegir la EC_A.

> **Def 14.2 [Perfil de usuario]** De acuerdo a lo expuesto, consideramos que el perfil de un usuario está determinado por su grado de experiencia, tanto en la materia del actual sistema hipermedia como en la navegación de sistemas web en general. Además, de por el subdominio de conocimiento en el que dicho usuario está más interesado.

Por lo tanto las características del modelo de usuario que el sistema consulta para determinar el perfil del usuario y en orden a éste ofrecerle una determinada EC_A son las siguientes:

- Experiencia del usuario:

- Experiencia en la materia, es decir, el conocimiento general que tiene el usuario sobre los conceptos tratados en el sistema hipermedia. En el modelo de usuario, esta característica es notada como **experiencia-materia** y representada mediante los grados: 0(“nulo”), 1(“bajo”), 2(“medio”), 3(“alto”), 4(“total”).

- Experiencia de navegación, esto es, hasta qué punto está el usuario familiarizado con el uso de sistemas hipermedia, y similares. Esta característica se nota como **experiencia-navegación** y es representada del mismo modo que la anterior.

- Subdominio de interés: Indica la parte del dominio de conocimiento del sistema hipermedia que despierta un mayor interés en el usuario. En el modelo de usuario se nota como **subdominio-de-interés** y su valor debe coincidir con alguno de los subdominios de conocimiento definidos por el autor sobre la EC_M que recoge el dominio completo.

2.2 Etiquetas para una EC_A: valores y significado

Para que el sistema pueda decidir sobre qué estructura debe ofrecer a cada usuario basándose en las tres características mencionadas anteriormente, el autor tan sólo debe etiquetar cada estructura conceptual de aprendizaje con esos mismos tres atributos: experiencia en la materia, experiencia de navegación y subdominio de conocimiento.

Quedando el etiquetado de una estructura EC_A con la forma general que se indica en la ecuación 2.

$$\text{etiquetado}(EC_A) = \{(\text{experiencia-materia}, \text{etiqueta}^1), (\text{experiencia-navegación}, \text{etiqueta}^2), (\text{subdominio-de-conocimiento}, \text{etiqueta}^3)\} \quad (2)$$

En la tabla 1 se establece el valor que puede tener la etiqueta de cada uno de los tres atributos o características requeridas en el etiquetado de una EC_A.

Tabla 1: Etiquetado de una EC_A

i	atributo^i	etiqueta^i
1	experiencia-materia	Es un intervalo de grados de experiencia, con la forma: $[et_{SEM}^1, et_{SEM}^2]$ donde valor-numérico(et_{SEM}^1) ≤ valor-numérico(et_{SEM}^2).
2	experiencia-navegación	
3	subdominio-de-conocimiento	Es un conjunto no vacío de parejas: $(\text{subdominio}^j, \text{porc}^j)$ donde subdominio^j es un subdominio de conocimiento válido y porc^j es un valor porcentual. Esto es, $0 < \text{porc}^j \leq 1$.

Como puede observarse en la tabla 1, para los dos atributos de tipo experiencia, el valor asignado en el etiquetado consiste en un intervalo que recoge el grado o grados de experiencia para los que es adecuada la estructura considerada. Para ambos, el rango de experiencia asignado por defecto es el que abarca todos los niveles de experiencia, es decir, el intervalo [“nulo”, “total”].

La etiqueta del tercer atributo indica el subdominio o subdominios de conocimiento capturados en la presentación sobre la que se define la estructura conceptual de aprendizaje considerada. Para cada subdominio incluido, el autor debe establecer de forma aproximada la porción del mismo, expresada con un porcentaje, que está visible en dicha presentación. En caso de no hacerlo el sistema presupone un 100%.

En la tabla 2 se indica, en función del valor asignado a cada etiqueta, para qué conjunto de usuarios es adecuada la EC_A, describiendo por tanto el perfil de usuario al que está dirigida.

Tabla 2: Perfil adecuado para una EC_A

Etiquetado(EC_A)	EC_A adecuada para:
experiencia-navegación = $[et_{SEM}^1, et_{SEM}^2]$	Un usuario cuyo grado de experiencia en el tipo de experiencia considerado es et_{SEM}^u, siendo: $et_{SEM}^1 \leq et_{SEM}^u \leq et_{SEM}^2$. Mejor cuanto más se aproxime et_{SEM}^u al centro del intervalo.
experiencia-materia = $[et_{SEM}^1, et_{SEM}^2]$	
subdominio-de-conocimiento = $\{(\text{subdominio}^1, \text{porc}^1), (\text{subdominio}^2, \text{porc}^2), \ldots, (\text{subdominio}^n, \text{porc}^n)\}$	Un usuario cuyo subdominio-de-interés es alguno de los especificados. Esto es: $\exists_{i=1..n}$ tal que: subdominio-de-interés = subdominio^i Mejor cuanto mayor sea el porcentaje capturado para dicho subdominio, esto es, porc^i.

En la tabla se refleja cómo una EC_A es adecuada para un usuario, si el subdominio de interés especificado por éste pertenece al conjunto de subdominios capturados en dicha estructura. Obviamente una EC_A es más adecuada a medida que el porcentaje recogido del subdominio de interés del usuario es mayor.

Por ejemplo, si el subdominio de interés del usuario es "x", una estructura de aprendizaje EC_A^1 es más adecuada respecto a este atributo que otra EC_A^2, si en la presentación sobre la que se define EC_A^1 se captura un 70% del subdominio "x", mientras que en la presentación asociada a EC_A^2 tan sólo un 45%.

Por su parte, el nivel de experiencia adecuado para recorrer una EC_A es todo aquel que pertenezca al intervalo especificado en el etiquetado de dicha estructura para ese tipo de experiencia. Es decir, una estructura etiquetada con el intervalo [et_{SEM}^1, et_{SEM}^2] en el atributo experiencia de navegación, es adecuada para un usuario con grado de experiencia de navegación igual a et_{SEM}^u siempre que se cumpla que $et_{SEM}^u \geq et_{SEM}^1$ y $et_{SEM}^u \leq et_{SEM}^2$.

Tampoco, en este caso, todos los grados contenidos en un intervalo son igual de apropiados. Los niveles muy próximos a los límites del intervalo suelen ser menos adecuados que los más centrados, especialmente en intervalos amplios. Así, dado un intervalo [et_{SEM}^1, et_{SEM}^2] el nivel más adecuado, al que denominamos **grado óptimo de experiencia** ($G_{op\text{-}exp}$), es aquel que se encuentra en el centro del intervalo. Obviamente, cuando el intervalo comprende un número par de grados de experiencia son dos, en lugar de uno, los grados de experiencia situados en el centro del intervalo, siendo considerados grados igualmente óptimos ($G_{op1\text{-}exp}$, $G_{op2\text{-}exp}$).

Por ejemplo, el intervalo por defecto ["nulo", "total"] contiene cinco grados distintos y presenta un único valor central, siendo el grado más adecuado el expresado por la etiqueta semántica "medio".

El intervalo ["bajo","total"] contiene cuatro grados de experiencia, y por lo tanto, dos valores centrales e igual de apropiados: "medio" y "alto".

A continuación se incluye la fórmula utilizada por el sistema para obtener el grado o grados óptimos de un intervalo. Esta operación es realizada automáticamente para cada estructura de aprendizaje y cada etiqueta de tipo intervalo.

Paso 1. Calcular la longitud (L) del intervalo [et_{SEM}^1, et_{SEM}^2], esto es, el número de grados que incluye.

L = [valor-numérico(et_{SEM}^2) – valor-numérico(et_{SEM}^1)] + 1

Paso 2a. Si L es un número impar entonces:

$$G_{op-\exp} = \frac{L-1}{2} + valor - numérico\left(et_{SEM}^{\ 1}\right)$$

Paso 2b. Si L es un número par entonces:

$$G_{op1-\exp} = ParteEntera\left(\frac{L-1}{2}\right) + valor - numérico\left(et_{SEM}^{\ 1}\right)$$

$$G_{op2-\exp} = \left(ParteEntera\left(\frac{L-1}{2} \right) + 1 \right) + valor - numérico\left(et_{SEM}{}^{1} \right)$$

Algoritmo 1. Calcular el grado óptimo G_{op}

2.3 Criterios para etiquetar

Como se ha comentado en secciones anteriores, es responsabilidad del autor etiquetar cada estructura conceptual de aprendizaje. Evidentemente, los criterios que sigue el autor para dar valor a cada etiqueta son personales. No obstante, en la sección actual se esbozan algunos criterios que sería sensato tener en cuenta a la hora de establecer las etiquetas experiencia-materia y experiencia-navegación de una estructura de aprendizaje, definida como $EC_A{}^j = (EC_M, Rw, MU, EC_P{}^k, RTnb^i, Ro^i, Ru^j, Rk^j)$.

- **Experiencia en la materia:**

El intervalo de grados de experiencia en la materia al que se enfoca la $EC_A{}^j$ debe ser más alto a medida que la presentación implicada, esto es $EC_P{}^k$, contenga mayor número de ítems catalogados con dificultad elevada. Del mismo modo, si los conceptos incluidos en dicha presentación son complejos, compuestos, abstractos o muy técnicos, el intervalo de experiencias requerido para utilizar esa estructura de aprendizaje debe ser alto.

Por su parte, un conjunto de reglas de conocimiento Rk^j poco restrictivas y reglas de actualización Ru^j generosas requiere una experiencia alta en la materia. Mientras que las reglas de conocimiento restrictivas y las reglas de actualización moderadas son más adecuadas para evitar problemas de comprensión en usuarios con escasa experiencia en la materia.

- **Experiencia de navegación:**

Obviamente, la cantidad de elementos mostrados en la $EC_P{}^k$ debe influir en el perfil de usuario al que va destinada la estructura de aprendizaje $EC_A{}^j$. Concretamente, las presentaciones con un número elevado de conceptos, ítems y asociaciones y, en general, las presentaciones extensas son más difíciles de navegar y, por lo tanto, requieren una experiencia de navegación mayor.

El perfil de usuario para el que es apropiada una $EC_A{}^j$ depende también de las reglas de orden Ro^i y navegabilidad $RTnb^i$ asociadas. Ya que, por ejemplo, si el autor extiende la navegación de la mayoría de las relaciones conceptuales existentes en $EC_P{}^k$, está aumentando las posibilidades de navegación y consecuentemente la probabilidad de que usuarios con poca experiencia de navegación se sientan perdidos o desorientados.

A partir de las directrices expuestas, se deduce que los intervalos de experiencia que debe asignar el autor a una estructura de aprendizaje no dependen de un único factor, sino de la combinación de muchos de ellos. Es necesario tener en cuenta: el subconjunto de ítems, conceptos y relaciones incluidas en la presentación $EC_P{}^k$ y el conjunto de reglas de orden, navegabilidad, conocimiento y actualización definidas sobre ésta en la estructura actual.

- **Subdominio de conocimiento:**

Por el contrario, el subdominio o los subdominios capturados en una estructura de aprendizaje EC_A^j dependen únicamente de la presentación EC_P^k sobre la que ésta se define. Entonces, ¿tiene el autor que dar valor a la etiqueta subdominio-de-conocimiento en cada estructura de aprendizaje? La respuesta es, por supuesto, negativa. El autor sólo debe dar valor a la etiqueta subdominio-de-conocimiento una vez para cada presentación existente.

Después, para cada estructura de aprendizaje, el valor de la etiqueta subdominio-de-conocimiento es tomado de la presentación asociada. De forma que dos estructuras de aprendizaje EC_A^j, EC_A^s definidas sobre la misma presentación EC_P^k siempre tienen idéntico valor en esta etiqueta, sin necesidad de que el autor lo repita en cada una de ellas. Esto es, subdominio-de-conocimiento(EC_A^j) = subdominio-de-conocimiento(EC_A^s) = subdominio-de-conocimiento(EC_P^k).

Por ejemplo, un posible etiquetado para una estructura de aprendizaje identificada como EC_A^1 es el que se muestra en la ecuación 3.

etiquetado(EC_A^1) = { (experiencia-materia, [“medio”,”alto”]), (experiencia-navegación, [“nulo”, “medio”]), (subdominio-de-conocimiento, {(adaptación al usuario, 75%), (modelo SEM-HP, 40%)}) } (3)

En la tabla 3 se muestra por separado el valor de cada etiqueta y junto a las dos primeras el grado óptimo calculado.

Tabla 3: Etiquetado[EC_A^1]

experiencia-materia(EC_A^1)	[“medio”,”alto”]	$G_{op1\text{-}expM}$ = 2 (“medio”) $G_{op2\text{-}expM}$ = 3 (“alto”)
experiencia-navegación(EC_A^1)	[“nulo”, “medio”]	$G_{op\text{-}expN}$ = 1(“bajo”)
subdominio-de-conocimiento(EC_A^1)	{(adaptación al usuario, 75%), (modelo SEM-HP, 40%)}	

En dicho etiquetado se establece que, a juicio del autor:

a) La presentación sobre la que se define EC_A^1 captura dos subdominos de conocimiento: aproximadamente un 75% del subdominio denominado “adaptación al usuario” y un 40% del subdominio “modelo SEM-HP”,

b) La estructura de aprendizaje está especialmente dirigida a usuarios con una experiencia “media” o “alta” en la materia, y

c) Se trata de una estructura adecuada para usuarios con poca experiencia de navegación (“nulo”, “bajo” o “medio”), siendo “bajo” el grado de experiencia más adecuado.

Se ha comentado en la sección anterior, que las etiquetas experiencia-materia y experiencia-navegación tienen por defecto el valor [“nulo”, “total”], lo que indica que son adecuadas para cualquier usuario, independientemente de su grado de experiencia. Pero, ¿cuál es el valor por defecto para la etiqueta subdominio-de-conocimiento en una presentación? Dicho valor no puede ser otro que “indiferente”, lo que significa que el

subdominio de conocimiento capturado es desconocido, por lo que las estructuras de aprendizaje definidas sobre esa presentación sólo son adecuadas para los usuarios que no han especificado interés especial por ningún subdominio, esto es, aquellos que precisamente tienen el valor "indiferente" en el atributo subdominio-de-interés de su modelo de usuario.

2.4 Evolución del etiquetado

En definitiva, dejando de lado el etiquetado por defecto, el autor debe asignar valor a la etiqueta subdominio-de-conocimiento cada vez que cree una nueva estructura conceptual de presentación. De igual modo debe establecer el valor de las etiquetas experiencia-materia y experiencia-navegación cada vez que defina una estructura conceptual de aprendizaje.

Tal y como se perseguía, el proceso de etiquetado es una tarea sencilla, que requiere poco esfuerzo por parte del autor. Pero, ¿puede el autor modificar el etiquetado de una estructura? Ineludiblemente, la respuesta a esta pregunta debe ser sí. El modelo permite hacer evolucionar los distintos elementos del sistema hipermedia. Lo que da lugar a que, a veces, cuando cambian las características de una estructura también sea necesario modificar el etiquetado de la misma.

Con el objetivo de hacer el etiquetado de las estructuras una tarea flexible y abierta a cambios, el sistema proporciona un conjunto de acciones evolutivas que el autor puede usar para modificar cómoda y consistentemente el etiquetado de sus presentaciones y estructuras de aprendizaje. Las acciones disponibles se describen y especifican en las secciones 2.4.1 a 2.4.5.

Algunas de estas acciones controlan la confección y mantenimiento de una lista, a la que denominamos *LSubC*. Esta lista se compone de todos los subdominios de conocimiento existentes en el sistema hipermedia. Es decir, reúne el conjunto de subdominios en los que, según el autor, se puede dividir la estructura conceptual de memorización. *LSubC* es una lista dinámica, ya que no tiene que ser creada de antemano por el autor, sino que cada vez que éste etiqueta una presentación con un subdominio nuevo, éste es automáticamente incluido en *LSubC*. El contenido de esta lista se muestra al usuario para que seleccione su subdominio de interés de entre todos los posibles (véase el capítulo 12, Modelo de Usuario).

2.4.1 Añadir un subdominio al etiquetado de una presentación

Tabla 4: Acciones evolutivas para modificar el etiquetado de una EC_P -AñadirSubdominioEC_p-

ACe[1][Etiquetado]	**AñadirSubdominioEC_p(k, nombSubC, descSubC, porc): boolean;**
Argumentos	**k** es el índice que permite identificar la presentación cuyo etiquetado se desea modificar. Es decir, la presentación EC_p^k. **nombSubC** es el nombre del subdominio de conocimiento que se desea añadir al etiquetado de la presentación. **descSubC** es una breve descripción del subdominio. Sólo es necesario cuando es la primera vez que se utiliza.

	porc es un porcentaje que indica el contenido aproximado del subdominio *nombSubC* que está incluido en la presentación EC_p^k.
Definición	Esta acción evolutiva modifica la etiqueta subdominio-de-conocimiento asignada a la presentación EC_p^k, añadiendo un nuevo subdominio de conocimiento a la lista de subdominios capturados. Como consecuencia, el subdominio es también añadido en todas las estructuras de aprendizaje definidas sobre EC_P^k. Si el subdominio es nuevo, también se añade, junto con su descripción, a la lista de subdominios *LSubC*.
Precondición	Existe la presentación EC_p^k cuyo etiquetado se desea modificar.
	La etiqueta subdominio-de-conocimiento de dicha presentación no incluye actualmente el subdominio que se desea añadir. Esto es, (*nombSubC*, p) $\notin$ subdominio-de-conocimiento(EC_p^k) para ningún valor de p.
	El porcentaje indicado es válido. Esto es, 0 < *porc* ≤1.
	Si el subdominio no ha sido definido anteriormente, la descripción debe ser distinta de vacío. Esto es: Si *nombSubC* $\notin$ *LSubC* entonces *descSubC* ≠ “”.
Efecto	Se incluye una nueva pareja compuesta por el subdominio *nombSubC* y el porcentaje *porc*, en la etiqueta subdominio-de-conocimiento asociada a la presentación EC_p^k. subdominio-de-conocimiento’(EC_p^k) = subdominio-de-conocimiento(EC_p^k) $\cup$ {(*nombSubC*, *porc*)}
Propagación interna	Si el subdominio añadido es inédito, es decir no se encuentra en la lista *LSubC*, es añadido a dicha lista con la descripción asociada. Esto es: *LSubC* = *LSubC* $\cup$ {(*nombSubC*, *descSubC*)}.
Propagación externa	El etiquetado subdominio-de-conocimiento es actualizado en todas las estructuras de aprendizaje EC_A^1,..., EC_A^m definidas a partir de la presentación EC_p^k. $\forall_{j=1..m}$ subdominio-de-conocimiento(EC_A^j) = subdominio-de-conocimiento(EC_p^k).
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que se ha modificado el etiquetado de la presentación satisfactoriamente.
Ejecutada_por	Autor.

2.4.2 Eliminar un subdominio del etiquetado de una presentación

Tabla 5: Acciones evolutivas para modificar el etiquetado de una EC_P -Eliminar$SubdominioEC_p$-

ACe^2[Etiquetado]	**Eliminar$SubdominioEC_p$(k, nombSubC): boolean;**
Argumentos	**k** es el índice que permite identificar la presentación EC_p^k cuyo etiquetado se desea modificar. **nombSubC** es el nombre del subdominio de conocimiento que se desea eliminar del etiquetado de la presentación.
Definición	Esta acción evolutiva modifica la etiqueta subdominio-de-conocimiento asignada a la presentación EC_p^k, borrando el subdominio de conocimiento denominado *nombSubC*. Como consecuencia inmediata, el subdominio es eliminado del etiquetado de todas las estructuras de aprendizaje definidas sobre EC_P^k. Si el subdominio no es capturado en ninguna otra presentación, es eliminado de la lista de subdominios *LSubC*.
Precondición	Existe la presentación EC_p^k cuyo etiquetado se desea modificar.
	La etiqueta subdominio-de-conocimiento de dicha presentación incluye el subdominio que se desea eliminar. Esto es, $\exists$p tal que: (*nombSubC*, p) $\in$ subdominio-de-conocimiento(EC_p^k)
Efecto	Se elimina la pareja correspondiente al subdominio *nombSubC* en la etiqueta subdominio-de-conocimiento de la presentación EC_p^k. subdominio-de-conocimiento'(EC_p^k) = subdominio-de-conocimiento(EC_p^k) - (*nombSubC*, p) Tras el paso anterior, si la etiqueta subdominio-de-conocimiento ha quedado vacía, se le asigna el valor por defecto ("indiferente" con porcentaje 100%). Esto es: Si subdominio-de-conocimiento'(EC_p^k) = $\varnothing$ entonces subdominio-de-conocimiento'(EC_p^k) = {("indiferente", 1)}.
Propagación interna	Si el subdominio eliminado únicamente aparecía en la presentación actual, es eliminado también de la lista de subdominios existentes. Esto es: Si $\forall_{j=1..n}$ (*nombSubC*, p) $\notin$ subdominio-de-conocimiento(EC_p^j), entonces *LSubC* = *LSubC* - (*nombSubC*, desc).
Propagación externa	El etiquetado subdominio-de-conocimiento es actualizado en todas las estructuras de aprendizaje EC_A^1,..., EC_A^m definidas a partir de la presentación EC_p^k. $\forall_{j=1..m}$ subdominio-de-conocimiento(EC_A^j) = subdominio-de-conocimiento(EC_p^k).
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y devuelve *False*, en otro caso devuelve *True* indicando que se ha modificado el etiquetado de la presentación satisfactoriamente.
Ejecutada_por	Autor.

2.4.3 Modificar el porcentaje de un subdominio en el etiquetado de una presentación

Tabla 6: Acciones evolutivas para modificar el etiquetado de una EC_P -ModificarSubdominioEC_p-

ACe^3[Etiquetado]	**ModificarSubdominioEC_p(k, nombSubC, porc'): boolean;**
Argumentos	**k** es el índice que permite identificar la presentación EC_p^k cuyo etiquetado se desea modificar. **nombSubC** es el nombre del subdominio de conocimiento para el que se desea cambiar el porcentaje capturado en EC_p^k. **porc'** es el nuevo porcentaje.
Definición	Esta acción evolutiva modifica la etiqueta subdominio-de-conocimiento asignada a la presentación EC_p^k, sustituyendo el porcentaje asociado al subdominio denominado *nombSubC* con el nuevo valor (*porc'*) que se pasa como argumento. Como consecuencia inmediata, el subdominio es modificado en todas las estructuras de aprendizaje definidas sobre EC_P^k.
Precondición	Existe la presentación EC_p^k cuyo etiquetado se desea modificar.
	La etiqueta subdominio-de-conocimiento de dicha presentación incluye el subdominio cuyo porcentaje se desea actualizar. Esto es, $\exists$porc tal que: (*nombSubC*, porc) $\in$ subdominio-de-conocimiento(EC_p^k)
	El porcentaje indicado es válido. Esto es, 0 < *porc'* ≤1.
Efecto	En la etiqueta subdominio-de-conocimiento asociada a EC_p^k, se sustituye el porcentaje porc actualmente asociado al subdominio *nombSubC* por el nuevo porcentaje *porc'*. Esto es: subdominio-de-conocimiento'(EC_p^k) = (subdominio-de-conocimiento(EC_p^k) - (*nombSubC*, porc)) $\cup$ {(*nombSubC*, *porc'*)}
Propagación externa	El etiquetado subdominio-de-conocimiento es actualizado en todas las estructuras de aprendizaje EC_A^1,..., EC_A^m definidas a partir de la presentación EC_p^k. $\forall_{j=1..m}$ subdominio-de-conocimiento(EC_A^j) = subdominio-de-conocimiento(EC_p^k).
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que se ha modificado el etiquetado de la presentación satisfactoriamente.
Ejecutada_por	Autor.

2.4.4 Modificar la experiencia en el etiquetado de una estructura de aprendizaje

Tabla 7: Acciones evolutivas para modificar el etiquetado de una EC_A -ModificarExperienciaEC_A-

ACe[4][Etiquetado]	**ModificarExperienciaEC_A(j, T, intervalo'$_{Exp}$): boolean;**
Argumentos	**j** es el identificador de la estructura conceptual de aprendizaje EC_A^j cuyo etiquetado se desea modificar. **T** especifica la etiqueta de tipo experiencia para la que se desea modificar su valor. Sólo admite dos valores: "M" y "N". De forma que T="M" indica el atributo experiencia-materia y T="N" experiencia-navegación. **Intervalo'$_{Exp}$** es el nuevo intervalo de grados de experiencia.
Definición	Esta acción evolutiva modifica el intervalo de grados de experiencia asignado a la etiqueta experiencia-navegación o la etiqueta experiencia-materia según el valor del argumento T.
Precondición	Existe la estructura de aprendizaje EC_A^j cuyo etiquetado se desea modificar.
	El nuevo intervalo de grados de experiencia es correcto. Esto es: intervalo'$_{Exp}$ = [et_{SEM}^1, et_{SEM}^2], donde valor-numérico(et_{SEM}^1) ≤ valor-numérico(et_{SEM}^2).
Efecto	En la etiqueta subdominio-de-conocimiento asociada a EC_A^j, se sustituye el intervalo de experiencia indicado en T por el que se especifica en el argumento intervalo'$_{Exp}$. Esto es: Si T = "M" entonces experiencia-materia'(EC_A^j) = intervalo'$_{Exp}$ Si T = "N" entonces experiencia-navegación'(EC_A^j) = intervalo'$_{Exp}$ Cualquiera que sea el valor de T se recalcula el grado o grados de experiencia óptimos para el nuevo intervalo (algoritmo 1).
Salida	Si no se satisface alguna de las precondiciones la acción evolutiva no se realiza y se devuelve *False*, en otro caso se devuelve *True* indicando que se ha modificado el etiquetado satisfactoriamente.
Ejecutada_por	Autor.

2.4.5 Resumen de acciones evolutivas para modificar el etiquetado

Tal y como se observa en la siguiente tabla, la ejecución de cualquiera de las acciones evolutivas anteriormente presentadas es responsabilidad exclusiva del autor, no pudiendo ser llevada a cabo de forma automática por el sistema. Dicho de otro modo, sólo el autor puede modificar el etiquetado de una presentación o estructura de aprendizaje.

No obstante, el sistema proporciona una serie de facilidades, como por ejemplo, la existencia de un etiquetado por defecto o el hecho de fijar automáticamente el valor de la etiqueta subdominio-de-conocimiento asignado a una presentación en todas las estructuras de aprendizaje derivadas.

Tabla 8: Acciones evolutivas para modificar el etiquetado

ACe[Etiquetado]	etiqueta modificada	ejecutada sobre	afecta a	
AñadirSubdominioEC$_P$	subdomino-de-conocimiento	Una presentación (EC$_P$)	EC$_P$ y todas las EC$_A$ asociadas	Ejecutada por Autor
EliminarSubdominioEC$_P$				
ModificarSubdominioEC$_P$				
ModificarExperienciaEC$_A$	experiencia-materia	Una estructura de aprendizaje (EC$_A$)	EC$_A$	
	experiencia-navegación			

En la tabla 8, también se observa, cómo el autor realiza el etiquetado del atributo subdominio-de-conocimiento desde la presentación (propagándose a todas las estructuras de aprendizaje asociadas), mientras que el etiquetado de los atributos experiencia-materia y experiencia-navegación debe ser realizado de forma específica para cada estructura de aprendizaje.

3. ELECCIÓN AUTOMÁTICA DE LA EC$_A$

Para determinar la estructura o estructuras de aprendizaje que mejor se ajustan al perfil de un determinado usuario, el sistema aplica una función de comparación que confronta los valores del usuario con el etiquetado de cada estructura. Esta comparación involucra a los tres atributos considerados en el etiquetado: subdominio-de-conocimiento, experiencia-materia y experiencia-navegación.

Para explicar el procedimiento de comparación, usaremos la notación *subdominio-de-interés(MU), experiencia-materia(MU)* y *experiencia-navegación(MU)* para reflejar los valores de esos tres atributos en el modelo del usuario actual.

Paso 1. Se recorre el conjunto de todas las estructuras de aprendizaje, EC_A^1, EC_A^2,..., EC_A^m, existentes en el sistema actual. Creando un subconjunto ***S*** donde sólo se incluyen las EC_A^j que en la etiqueta subdominio-de-conocimiento contienen el subdominio de interés especificado por el usuario.

A) Cuando el subdominio de interés del usuario es "indiferente" todas las estructuras de aprendizaje son seleccionadas. Esto es:

Si subdominio-de-interés(MU) = "indiferente" entonces S = {EC_A^1, EC_A^2,..., EC_A^m}. Es decir, $\forall_{j=1..m}\ EC_A^j \in S$.

B) En otro caso (subdominio-de-interés(MU) ≠ "indiferente"):

Dado subdominio-de-interés(MU) = nombSubC, la estructura de aprendizaje EC_A^j es seleccionada, es decir $EC_A^j \in S$, sólo si $\exists p$ tal que (nombSubC, p) $\in$ subdominio-de-conocimiento(EC_A^j).

Paso 2. Para cada estructura EC_A^j en el conjunto S, se obtienen tres indicadores: **ISubC**, **IExpM** e **IExpN**, asociados respectivamente a las etiquetas subdominio-de-conocimiento, experiencia-materia y experiencia-navegación.

Cada indicador establece el grado en que el valor de esa etiqueta en Etiquetado(EC_A^j) se aproxima al valor que para ese mismo atributo presenta actualmente el modelo del usuario.

Paso 2.1 El indicador **ISubC(EC_A^j)** refleja lo adecuada que es la estructura EC_A^j de acuerdo al subdominio de interés especificado por el usuario. El valor de este indicador se corresponde con el porcentaje de dicho subdominio que es capturado en la estructura de aprendizaje considerada.

ISubC(EC_A^j) = porc, donde (nombSubC, porc) $\in$ subdominio-de-conocimiento(EC_A^j) y nombSubC = subdominio-de-interés(MU).

Paso 2.2 El indicador **IExpM(EC_A^j)** refleja cómo de apropiada es la estructura de aprendizaje EC_A^j respecto al grado de experiencia en la materia que posee actualmente el usuario. Para calcular este indicador el sistema compara el grado de experiencia en la materia del usuario, experiencia-materia(MU), con el grado o grados óptimos asociados al intervalo de experiencia que da valor a la etiqueta experiencia-materia(EC_A^j). Ver el cálculo del indicador en la tabla 9.

Paso 2.3 El indicador **IExpN(EC_A^j)** indica cómo de adecuada es la estructura de aprendizaje EC_A^j basándose en el grado de experiencia de navegación del usuario. Para calcular este indicador el sistema compara el valor de la etiqueta experiencia-navegación(EC_A^j) con el grado de experiencia de navegación almacenado en el modelo de usuario, esto es experiencia-navegación(MU). Ver el cálculo del indicador en la tabla 10.

Algoritmo 2(continua...). Elegir la EC_A

El calculo de los indicadores IExpM(EC_A^j) e IExpN(EC_A^j) se detalla en las tablas 9 y 10 respectivamente, y puede ser descrito de la siguiente forma:

- El indicador vale 1 si el grado de experiencia del usuario coincide con el grado óptimo ($G_{op\text{-}exp}$) del intervalo que etiqueta la EC_A^j.

- En otro caso se obtiene la longitud *L* del intervalo, y la distancia absoluta *d* que existe entre el grado de experiencia del usuario y el grado óptimo. Si hay dos óptimos se trabaja con el valor medio de ambos.

- Si el grado de experiencia del usuario pertenece al intervalo, el indicador se calcula como $(L - d)/L$. Este indicador es siempre positivo, puesto que la distancia *d* no puede ser mayor que la longitud del intervalo *L*. Además, cuanto más se aleja la experiencia del usuario del grado óptimo (mayor valor para *d*) menor es el numerador de la división $(L\text{-}d)$ y, por lo tanto, más pequeño el indicador obtenido. El valor es 1 sólo si $d = 0$.

- Si el grado de experiencia del usuario queda fuera del intervalo, el indicador es negativo. Para obtenerlo se divide la distancia del usuario al grado óptimo (*d*) entre la distancia máxima posible al grado óptimo (d_{max}) y se hace negativo el resultado. Usando esta fórmula: $-(d/d_{max})$ cuanto más se aproxima la distancia del usuario al óptimo a la distancia máxima ($d \approx d_{max}$), más grande en valor negativo es el indicador que se obtiene. El valor es -1 sólo si $d = d_{max}$.

- Puesto que el intervalo más amplio posible es [0,4] si el grado óptimo $G_{op\text{-}exp}$ es 0, 1 o 2 la distancia máxima es $4\text{-}G_{op\text{-}exp}$, y si $G_{op\text{-}exp}$ es 2, 3 o 4 la distancia máxima es $G_{op\text{-}exp}$. Por lo tanto, de forma general d_{max} = máximo($G_{op\text{-}exp}$, $4\text{-}G_{op\text{-}exp}$).

Tabla 9: Cálculo del indicador IExpM(EC_A^j)

experiencia-materia(EC_A^j) = [et_{SEM}^1, et_{SEM}^2] con un grado óptimo $G_{op\text{-}expM}$
Si experiencia-materia(MU) = $G_{op\text{-}expM}$ entonces IExpM(EC_A^j) = 1 En otro caso: L = valor-numérico(et_{SEM}^2) – valor-numérico(et_{SEM}^1) d = | experiencia-materia(MU) – $G_{op\text{-}expM}$ | Si $et_{SEM}^1 \leq$ experiencia-materia(MU) $\leq et_{SEM}^2$ entonces IExpM(EC_A^j) = $\frac{L-d}{L}$ Si experiencia-materia(MU) $\leq et_{SEM}^1$ o experiencia-materia(MU) $\geq et_{SEM}^2$ entonces IExpM(EC_A^j) = $-\frac{d}{\max(G_{op-\exp M}, 4-G_{op-\exp M})}$
experiencia-materia(EC_A^j) = [et_{SEM}^1, et_{SEM}^2] con dos grados óptimos $G_{op1\text{-}expM}$ y $G_{op2\text{-}expM}$
Si experiencia-materia(MU) = $G_{op1\text{-}expM}$ o experiencia-materia(MU) = $G_{op2\text{-}expM}$ entonces IExpM(EC_A^j) = 1 En otro caso: L = valor-numérico(et_{SEM}^2) – valor-numérico(et_{SEM}^1) $G_{op\text{-}expM}$ = $\frac{L}{2} + valor-numérico(et_{SEM}^1)$ d = | experiencia-materia(MU) – $G_{op\text{-}expM}$ | Si $et_{SEM}^1 \leq$ experiencia-materia(MU) $\leq et_{SEM}^2$ entonces IExpM(EC_A^j) = $\frac{L-d}{L}$ Si experiencia-materia(MU) $\leq et_{SEM}^1$ o experiencia-materia(MU) $\geq et_{SEM}^2$ entonces IExpM(EC_A^j) = $-\frac{d}{\max(G_{op-\exp M}, 4-G_{op-\exp M})}$

Tabla 10: Cálculo del indicador IExpN(EC_A^j)

experiencia-navegación(EC_A^j) = [et_{SEM}^1, et_{SEM}^2] con un grado óptimo $G_{op\text{-}expN}$
Si experiencia-navegación(MU) = $G_{op\text{-}expN}$ entonces IExpN(EC_A^j) = 1 En otro caso: L = valor-numérico(et_{SEM}^2) – valor-numérico(et_{SEM}^1) d = | experiencia-navegación(MU) – $G_{op\text{-}expN}$ | Si et_{SEM}^1 ≤ experiencia-navegación(MU) ≤ et_{SEM}^2 entonces IExpN(EC_A^j) = $\frac{L-d}{L}$ Si experiencia-navegación(MU) ≤ et_{SEM}^1 o experiencia-navegación(MU) ≥ et_{SEM}^2 entonces IExpN(EC_A^j) = $-\frac{d}{\max(G_{op-\exp N}, 4-G_{op-\exp N})}$
experiencia-navegación(EC_A^j) = [et_{SEM}^1, et_{SEM}^2] con dos grados óptimos $G_{op1\text{-}expN}$ y $G_{op2\text{-}expN}$
Si experiencia-navegación(MU) = $G_{op1\text{-}expN}$ o experiencia-navegación(MU) = $G_{op2\text{-}expN}$ entonces IExpN(EC_A^j) = 1 En otro caso: L = valor-numérico(et_{SEM}^2) – valor-numérico(et_{SEM}^1) $G_{op\text{-}expN} = \frac{L}{2} + valor-numérico(et_{SEM}^1)$ d = | experiencia-navegación(MU) – $G_{op\text{-}expN}$ | Si et_{SEM}^1 ≤ experiencia-navegación(MU) ≤ et_{SEM}^2 entonces IExpN(EC_A^j) = $\frac{L-d}{L}$ Si experiencia-navegación(MU) ≤ et_{SEM}^1 o experiencia-navegación(MU) ≥ et_{SEM}^2 entonces IExpN(EC_A^j) = $-\frac{d}{\max(G_{op-\exp N}, 4-G_{op-\exp N})}$

Aplicando lo establecido en las tablas 9 y 10, el indicador de experiencia es 1 cuando el nivel de experiencia del usuario coincide con el punto central del intervalo de experiencia que etiqueta la estructura de aprendizaje EC_A^j. Este indicador es menor que 1 (mayor que ½) si el grado de experiencia del usuario no es el óptimo pero se encuentra dentro del intervalo de experiencia adecuado para EC_A^j. Por supuesto, dicho número es

mayor cuanto más se acerca el grado de experiencia del usuario al grado óptimo. Por el contrario, el indicador es un número negativo mayor o igual que -1 cuando el grado de experiencia del usuario está fuera del intervalo especificado. Dicho número es más negativo conforme la experiencia del usuario está más lejos del grado óptimo, siendo -1 si se trata del valor más alejado posible.

Paso 3. A partir de S se crea un subconjunto **P** que contiene todas las estructuras evaluadas con indicadores de experiencia IExpM y IExpN positivos. Este subconjunto, contiene por tanto, las estructuras de aprendizaje adecuadas al nivel de experiencia del usuario. Esto es:

$EC_A^j \in P$ si $EC_A^j \in S$, cumpliendo que $IExpM(EC_A^j) > 0$ y $IExpN(EC_A^j) > 0$

Si el conjunto S está vacío, S = ∅, ir al Paso 5, en otro caso continuar con el Paso 4.

Paso 4. Si el conjunto P no está vacío se calcula un indicador total para cada estructura incluida en él. Este indicador es notado **$Itotal(EC_A^j)$** y refleja el grado con que dicha estructura es apropiada para el usuario.

Para obtener el indicador se hace la media de los tres indicadores parciales ISubC, IExpM e IExpN. Obviamente, se elige para el usuario la estructura de aprendizaje cuyo indicador total sea mayor.

$$\forall EC_A^j \in P \quad Itotal(EC_A^j) = \frac{ISubC\left(EC_A^{\ j}\right) + IExpM\left(EC_A^{\ j}\right) + IExpN\left(EC_A^{\ j}\right)}{3}$$

EC_A^j es elegida si $Itotal(EC_A^j)$ = máximo {$Itotal(EC_A^1)$, ..., $Itotal(EC_A^p)$} siendo EC_A^1 ,..., EC_A^p las estructuras de aprendizaje incluidas en P. Terminar.

Paso 5. Si el conjunto P es vacío se informa al usuario de que actualmente el sistema no dispone de ninguna estructura que capture el subdominio de conocimiento que le interesa y además sea adecuada para su nivel de experiencia.

Sin embargo si aún así, el usuario desea obtener una estructura para dicho subdominio, el sistema le proporciona la presentación que menos se aleje de su experiencia. Para ello calcula el indicador total de todas las presentaciones incluidas en S y devuelve aquella para la que se obtiene el valor mayor.

$$\forall EC_A^j \in S \quad Itotal(EC_A^j) = \frac{ISubC\left(EC_A^{\ j}\right) + IExpM\left(EC_A^{\ j}\right) + IExpN\left(EC_A^{\ j}\right)}{3}$$

EC_A^j es elegida si $Itotal(EC_A^j)$ = máximo{$Itotal(EC_A^1)$,..., $Itotal(EC_A^s)$} siendo EC_A^1 , ..., EC_A^s el conjunto de presentaciones incluidas en S.

Terminar.

Algoritmo 2(...continuación). Elegir la EC_A

A continuación se muestra un ejemplo donde puede verse la aplicación del procedimiento explicado. El ejemplo trata con tres estructuras diferentes: EC_A^1, EC_A^2 y EC_A^3 y determina cuál de ellas es más adecuada para un usuario que está interesado en navegar información acerca del subdominio "adaptación al usuario" y que posee bastante experiencia en la materia, pero tiene poca soltura en la navegación hipermedia.

La tabla 11 muestra el etiquetado correspondiente a cada una de las estructuras mencionadas, adjuntando junto a cada intervalo de experiencia, el grado o grados de experiencia óptimos.

Tabla 11: Etiquetado de tres estructuras de aprendizaje

	subdomino-de-conocimiento	experiencia-materia	experiencia-navegación
EC_A^1	{ (adaptación al usuario, 75%), (modelo SEM-HP, 40%) }	["medio","alto"] $G_{op1\text{-}expM}$ = 2 ("medio") $G_{op2\text{-}expM}$ = 3 ("alto")	["nulo", "medio"] $G_{op\text{-}expN}$ = 1 ("bajo")
EC_A^2	{(adaptación al usuario, 95%)}	["bajo", "alto"] $G_{op\text{-}expM}$ = 2 ("medio")	["nulo", "total"] $G_{op\text{-}expN}$ = 2 ("medio")
EC_A^3	{ (adaptación al usuario, 80%), (evolución, 30%) }	["medio", "total"] $G_{op\text{-}expM}$ = 3 ("alto")	["alto","total"] $G_{op1\text{-}expN}$ = 3 ("alto") $G_{op2\text{-}expN}$ = 4 ("total")

En la tabla 12 se muestra el resultado y los cálculos realizados para obtener los indicadores parciales y totales en cada estructura de acuerdo al perfil del usuario descrito: subdominio-de-interés(MU) = "adaptación al usuario", experiencia-materia(MU) = 3 ("alto") y experiencia-navegación(MU) = 1 ("bajo").

Tabla 12: Elección de la estructura adecuada para el usuario

	$\in$ S?	ISubC	IExpM	IExpN	$\in$ P?	Itotal
EC_A^1	Sí	0,75	1	1	Sí	$\frac{2,75}{3} \approx 0,91$
EC_A^2	Sí	0,95	$\frac{2-(\lvert 3-2\rvert)}{2} = \frac{1}{2} = 0,5$	$\frac{4-(\lvert 1-2\rvert)}{4} = \frac{3}{4} = 0,75$	Sí	$\frac{2,2}{3} \approx 0,73$
EC_A^3	Sí	0,80	1	$-\frac{(\lvert 1-3,5\rvert)}{\max(3,5,4-3,5)} \approx -0,71$	No	

Observe que inicialmente las tres estructuras son seleccionadas e incluidas en el conjunto *S*. Esto se debe a que todas ellas capturan, en mayor o menor grado, el subdominio de conocimiento "adaptación al usuario".

Después de obtener los indicadores parciales, sólo las dos primeras estructuras, EC_A^1 y EC_A^2, son incluidas en el conjunto *P*. EC_A^3 no pertenece a *P* porque su indicador IExpN(EC_A^3) es negativo, lo que significa que no es adecuada para la experiencia de navegación del usuario. Si observamos el valor de IExpN(EC_A^3) = -0,71, confirmamos que el grado de experiencia de navegación del usuario ("bajo"), está muy lejos de la experiencia necesaria para navegar esa presentación ("alto" o "total").

Finalmente la estructura más adecuada y, por lo tanto, la que se ofrece al usuario es EC_A^1, ya que su indicador total es mayor que el indicador total de EC_A^2.

CAPÍTULO 15

Modos de Navegación

Resumen

Todos los sistemas hipermedia adaptativos coinciden en personalizar el proceso de navegación del usuario, pero habitualmente lo hacen de acuerdo a un único modo de navegación prefijado. En este capítulo se describe la capacidad del Sistema de Aprendizaje para proporcionar cuatro modos de navegación alternativos, pudiendo el usuario elegir el que más le apetezca o convenga en cada momento. Se exponen las ventajas de navegar, en todos ellos, directamente sobre una red semántica. Y se establecen las similitudes y diferencias entre los modos, en cuanto a las restricciones de navegación, la gestión del modelo de usuario, la unidad de navegación y la adaptación realizada. Por último se describe el más básico de los cuatro modos de navegación: la navegación tradicional.

Tabla de contenidos

Modos de Navegación

1. La Estructura de Navegación

A menudo los enlaces que aparecen en un hiperdocumento son introducidos de forma caprichosa, casi aleatoria, y apenas muestran información sobre el destino del enlace y, mucho menos, sobre la relación semántica que éste guarda con el documento actual. Para evitar este tipo de problemas, por otro lado muy frecuentes en la creación de sistemas hipermedia, el modelo SEM-HP plantea una estructura de navegación donde la semántica es explícita [García, 00].

1.1 Una red semántica para navegar

La estructura de navegación propuesta en los capítulos anteriores constituye una red semántica (figura 1) donde se muestran los ítems de información disponibles, pero también el concepto o conceptos sobre los que éstos versan. Cada asociación entre un ítem y un concepto es etiquetada con un rol o función que concreta aún más el tipo de contenido del ítem: problema, introducción, antecedente, etc. Además, los conceptos implicados no aparecen desconectados e inconexos, sino relacionados semánticamente entre sí, lo que justifica su vínculo.

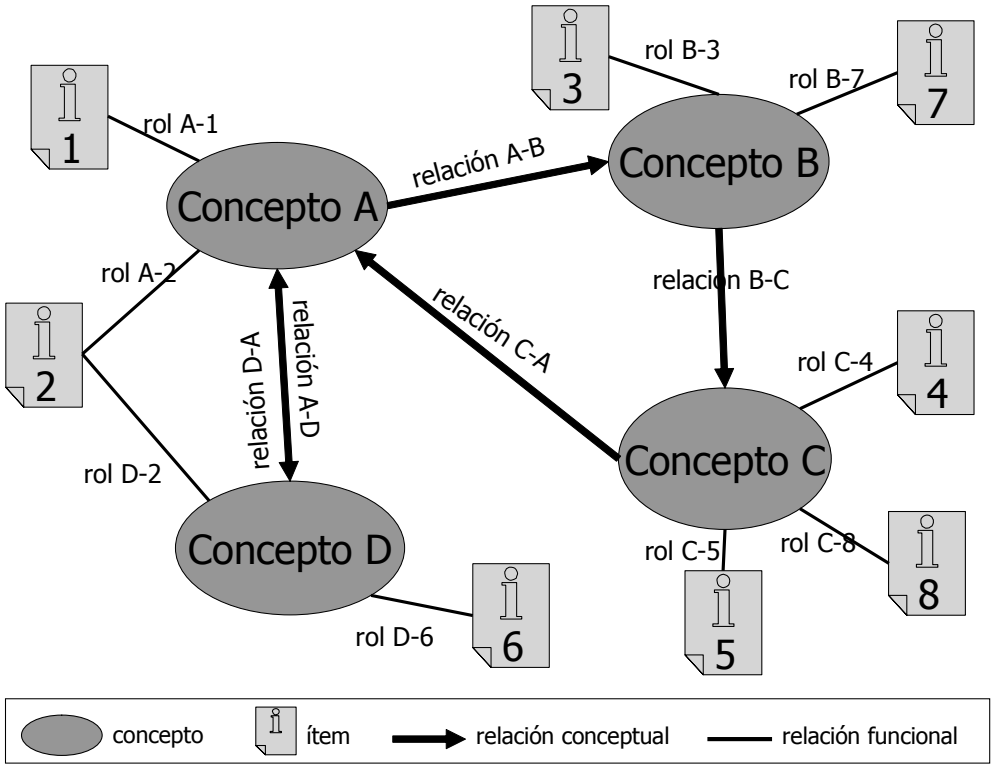


Figura 1: Una red semántica para navegar

En el capítulo anterior, se expuso el modo en que para cada usuario se elige la estructura de aprendizaje que mejor se ajusta a su perfil. Suponiendo que la estructura elegida es EC_A^j = (EC_M, Rw, MU, EC_P^i, $RTnb^k$, Ro^k, Ru^j, Rk^j), normalmente, la red semántica proporcionada al usuario es aquella que representa la estructura conceptual de

presentación EC_P^i. Decimos "normalmente" porque ya veremos que esto no es siempre así, ya que depende del modo de navegación que el usuario vaya a realizar.

De aquí en adelante nos referiremos como "estructura de navegación" al conjunto de ítems, conceptos y asociaciones (funcionales y conceptuales) existentes en la red semántica donde el usuario navega, esto es a la propia red. No debe confundirse este término con el de "estructura conceptual de navegación", ya que este último incluye además de la estructura conceptual de presentación, un conjunto de reglas de orden, Ro, y navebabilidad, RTnb, que restringen su utilización. Tal y como se ha indicado antes, tampoco en todos los casos, el término "estructura de navegación" coincide con el de "estructura conceptual de presentación".

1.2 Ventajas de la red semántica

Son muchos los sistemas que utilizan redes semánticas para estructurar el conocimiento que desean transmitir. RICH [Wang, 98], SCHOLAR [Carbonell, 70], ARCH [Winston, 75] o PEGASUS[Castell, 02b] son sólo algunos ejemplos. A diferencia del uso que se hace de éstas en SEM-HP, en la mayoría de los casos, las redes semánticas se emplean como representación interna pero no permiten una navegación explícita sobre ellas, presentan duras limitaciones en cuanto a las relaciones posibles y no están preparadas para evolucionar.

Por otro lado, cada vez son más los autores que apoyan el uso de este tipo de redes, sobre todo en ámbitos educativos. En concreto, Tirado Morueta [Tirado, 01] establece como aspecto crítico para el diseño de "hipermedias" en la enseñanza la utilización de redes semánticas que permitan organizar convenientemente los contenidos.

Una de las ventajas más importantes de la red semántica es su capacidad para auto-describirse. Es decir, si se conoce la representación utilizada, es posible obtener mucha información contemplando la estructura de navegación. En nuestro caso, como puede observarse en la leyenda de la figura 1, la representación utilizada para mostrar los conceptos, ítems y relaciones es básica e intuitiva.

Dado el ejemplo genérico de la figura 1 es inmediato detectar, entre otra, la siguiente información:

a) Que el ítem *i1* está asociado al concepto *A* con el rol *A-1*.

b) Que ligado a dicho concepto se encuentra otro ítem, denominado *i2*, el cuál también se asocia al concepto *D*. De modo que el rol que juega ese ítem para cada concepto, esto es *A-2* y *D-2*, puede coincidir, pero no tiene por qué ser así.

c) Que existe una estrecha conexión entre los conceptos *A* y *D* denotada por la relación bidireccional existente entre ellos. Por lo que, cabe esperar que a menudo sean visitados uno detrás del otro.

Toda la información semántica embebida en la propia estructura de navegación supone una mejora muy importante frente a un esquema de navegación donde sólo se visualice el contenido de la página seleccionada (figura 2). En este caso, el usuario es consciente de la existencia de otras páginas relacionadas únicamente a través de los enlaces que aparecen en la página actual. Para resaltar estas relaciones los enlaces suelen ser

marcados de forma especial dentro de la página, normalmente subrayados y con distinto color.

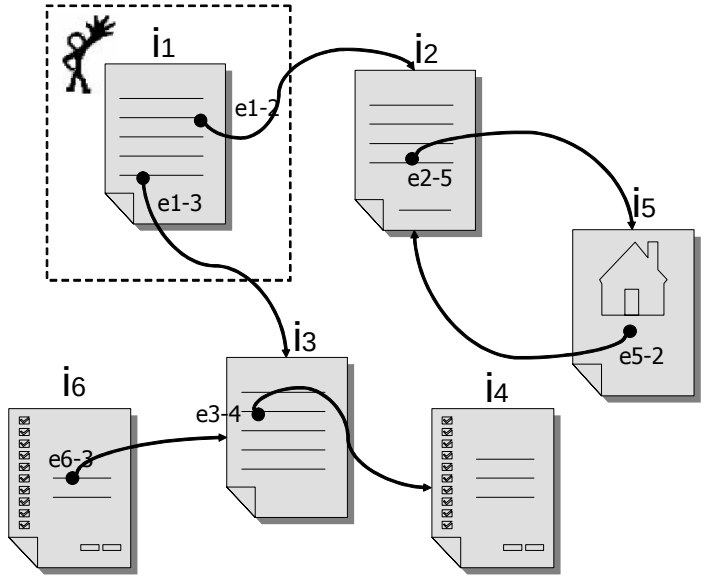


Figura 2: Navegación por páginas

Tomemos como ejemplo el conjunto de documentos de la figura 2 y los enlaces que en ella se muestran. Si la página actual es *i1*, el usuario únicamente ve su contenido, desde el que puede acceder a las páginas *i2* e *i3*, sin más información que la proporcionada en el anclaje de sus respectivos enlaces, esto es *e1-2* y *e1-3*. De este modo, el usuario desconoce la relación entre las página *i2* e *i5*, hasta que sigue el enlace *e1-2* que le lleva a la página *i2*.

Evidentemente, con la estructura de la figura 2 se pierde la visión de conjunto proporcionada por la red semántica. Es más fácil desorientarse, porque no se tiene delante el entorno de la página activa. Es decir, el usuario sabe a qué páginas puede llegar desde la página actual en un sólo paso. Pero no puede discernir el conjunto de páginas que llevan a ésta, ni las páginas alcanzables en dos o más pasos de navegación.

Para solventar el problema anterior, la mayoría de los sistemas hipermedia y sitios web incluyen un índice, menú o mapa de contenido, que facilita la navegación de la información proporcionada. El problema es que, a menudo, estos mapas no explicitan la relación semántica que existe entre sus entradas y presentan una estructura jerárquica.

Nosotros consideramos que la estructura de navegación debe ser una red y no un árbol. Los árboles establecen una taxonomía estricta, esto es, un concepto sólo puede tener relación con su padre y con sus hijos pero no con sus hermanos ni con conceptos de otros niveles. Sin embargo, existen multitud de relaciones conceptuales que no se ajustan a dicha estructura.

Por ejemplo, las relaciones conceptuales pueden ser cíclicas. No es de extrañar que un concepto *A* se relacione con un concepto *B* y éste con un concepto *C* que a su vez tiene alguna relación con el concepto *A* (tal y como aparece representado en la figura 1). En

caso de utilizar una estructura jerárquica, se haría necesario forzar o falsear las relaciones para evitar el ciclo.

En la figura 3 se muestra una estructura jerárquica obtenida a partir de la red semántica mostrada en la figura 1. Observe que la equivalencia no es total, es decir, al hacer la traducción se han perdido dos relaciones conceptuales: la que va desde el concepto *D* hasta el concepto *A* (*D*→*A*) y la que parte de *C* y termina en *A* (*C*→*A*).

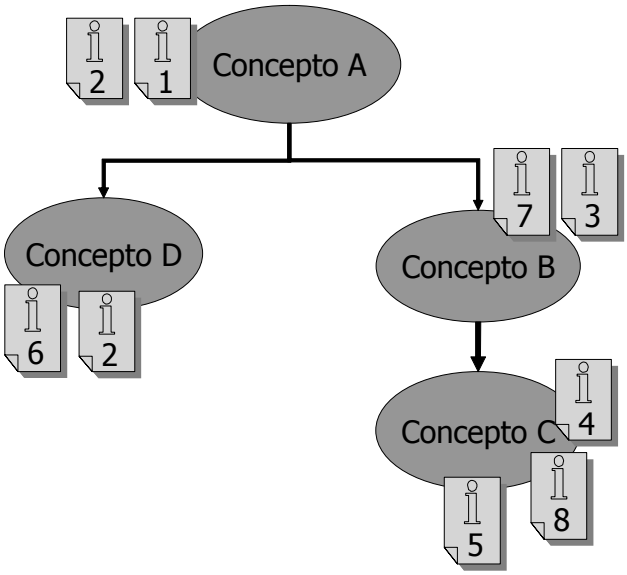


Figura 3: Navegación jerárquica

2. Información Contenida en la Red Semántica

En cualquier caso, ofrecer al usuario como estructura de navegación una red semántica, permite hacer explícito tanto el dominio conceptual como el dominio de información de la parcela de conocimiento que se representa. Esto tiene un claro beneficio, el usuario puede acceder a la información contenida en un ítem mediante su selección. Pero además, en todo momento, tiene a su alcance **información de contexto** acerca del ítem seleccionado y también del resto. Esta información adicional es fácilmente asimilada sin más que echar un rápido vistazo a la propia estructura de navegación. Y resulta muy útil, por ejemplo, para recordar cómo se ha llegado hasta el ítem actual o para decidir qué ítems consultar a continuación.

En la figura 4 se muestra una posible estructura de navegación, donde aparece resaltada la información de contexto para el ítem I9. Debajo, se incluye una leyenda, que explica el significado concreto de cada tipo de información. Observe cómo el usuario ve en todo momento el concepto al que se asocia el ítem actualmente seleccionado y su relación con otros conceptos, por lo que es más difícil que olvide el motivo por el que accedió a él, o en cualquier caso cómo desplazarse a otro concepto si éste ha dejado de interesarle. Por el contrario, si su deseo es obtener más información sobre el concepto actual sólo tiene que seleccionar otros ítems asociados a éste, pudiendo elegir entre uno u otro con ayuda del rol asociado al mismo.

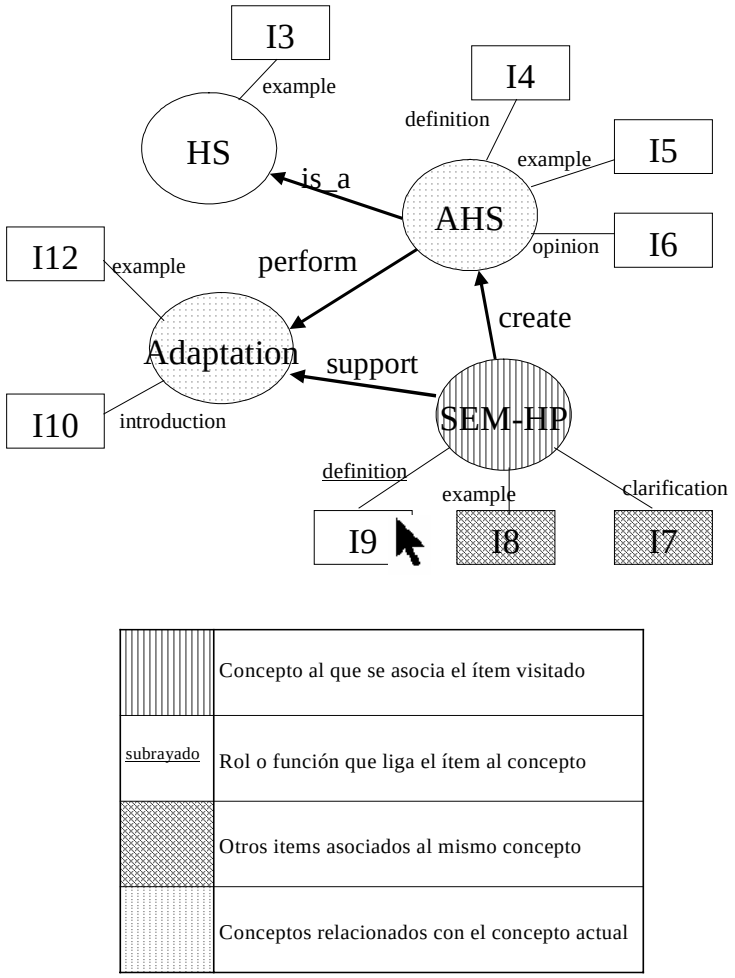


Figura 4: Información de soporte para el ítem I9

Toda esta información, intenta apoyar la navegación del usuario, en el sentido de reducir la posibilidad de desorientación y de ofrecerle una idea bastante cercana de cuál va a ser el contenido del ítem sin necesidad de inspeccionarlo. Esto permite reducir el número de veces que el usuario pierde el tiempo leyendo un ítem que no aborda la información que él esperaba, ya que de forma previa tiene información visual acerca del concepto sobre el que versa el ítem y el carácter con el que éste se trata: opinión, definición, ejemplo, comparación, etc.

3. DIVERSIDAD DE MODOS DE NAVEGACIÓN

El sistema de aprendizaje es capaz de realizar la **adaptación al usuario a varios niveles**. Ya hemos visto cómo inicialmente la estructura de aprendizaje es elegida en función del perfil del usuario. Más adelante veremos de qué manera la estructura de navegación es nuevamente personalizada mediante el empleo de distintas técnicas de adaptación (capítulos del 16 al 21). Este proceso de adaptación es individual para cada usuario, es decir, dista de un usuario a otro aunque ambos presenten el mismo perfil. Y depende del estado de conocimiento del usuario, de sus preferencias, intereses, y por supuesto, del proceso de navegación que realice.

Con respecto a este último punto, el sistema permite al usuario establecer la forma en que desea recorrer la información ofrecida [Medina, 04]. Así, el usuario elige el subdominio de conocimiento que desea recorrer y en qué modo. Dependiendo del **modo de navegación** que elija, se impondrán unas restricciones de navegación u otras, variando también las técnicas de adaptación empleadas.

Por lo tanto, el sistema no sólo adapta la navegación del usuario sino que, además, pone a su disposición distintos esquemas o modos de navegación, para que en cada momento ejercite el que más le convenga. El objetivo es siempre el mismo, que el usuario se encuentre cómodo utilizando un sistema que se ajusta en gran medida a sus necesidades.

Según la aplicación del sistema, esta capacidad podría estar deshabilitada. Por ejemplo, en sistemas hipermedia educacionales donde el tutor quiere garantizar que los alumnos solo acceden a documentos que están capacitados para entender. En cualquier caso, esta pérdida de control del usuario siempre debe ser solicitada por el autor, en su defecto cada usuario podrá elegir la manera en que desea recorrer la información.

Concretamente, el sistema contempla **cuatro modos de navegación** que se resumen muy brevemente en la tabla 1, y que serán explicados con más detalle en este y los sucesivos capítulos.

Tabla 1: Modos de navegación

	Tipo	Descripción	Restricciones	Unidad de navegación
LIBRE	**Tradicional**	Todos los ítems de la estructura de navegación son accesibles para el usuario.	Ninguna	Ítem
	Por Conceptos	Para cada concepto se compone un resumen con los ítems asociados a él.	Ninguna	Concepto
RESTRINGIDA	**Por Relación conceptual**	La navegación del usuario debe ajustarse a los enlaces de navegación existentes entre los conceptos.	Reglas de orden y navegabilidad	Ítem
	Por Conocimiento	Solo son accesibles los ítems para cuya comprensión el usuario está preparado.	Reglas de conocimiento	Ítem

Esta diversidad de formas de navegación [Medina, 03c] favorece la personalización, es decir, cada usuario preferirá un modo de navegación sobre otro, en función de cuál sea su estado de conocimiento, experiencia y preferencias. Por ejemplo, la navegación restringida por relación conceptual puede resultar pedagógica para un usuario con poca experiencia de navegación mientras que puede ser excesivamente restrictiva para uno más experto.

Además, permite tener en cuenta las circunstancias actuales del usuario. Esto es, un usuario puede elegir el tipo de navegación que mejor se ajuste al fin que persigue en un momento dado. Por ejemplo, para conocer exhaustivamente una parcela de conocimiento en la que es inexperto el usuario agradecerá la orientación y guía de la navegación por conocimiento. Mientras que para consultar un ítem concreto preferirá la navegación tradicional que le facilita el acceso rápido y directo al ítem sin ningún tipo de prerrequisito.

4. SIMILITUDES Y PARTICULARIDADES DE LOS MODOS DE NAVEGACIÓN

Tal y como se observa en la tabla 1, los cuatro modos de navegación disponibles pueden ser agrupados en dos tipos: navegación libre y navegación restringida.

4.1 Respecto a las restricciones de navegación

En la **navegación libre** se incluyen el modo de *navegación tradicional* y el modo de *navegación por conceptos*. El término libre se utiliza en el sentido de "sin restricciones", lo que implica que todas las unidades de navegación son accesibles, en cualquier momento, para el usuario.

> **Def 15.1 [Navegación libre]** En la navegación libre, el usuario puede seleccionar cualquiera de las unidades de navegación que aparecen en la estructura actual, es decir, no tiene que satisfacer ningún prerrequisito previo para poder acceder a la información ofrecida.

La diferencia entre los dos modos de navegación libre es la unidad de navegación, el ítem para la navegación tradicional y el concepto para la navegación por conceptos. En este último modo, el usuario selecciona unidades conceptuales en lugar de ítems de información. Por supuesto, cuando selecciona un concepto obtiene un compendio de la información asociada a éste.

Dentro de la **navegación restringida** se engloban el modo de *navegación por relación conceptual* y el modo de *navegación por conocimiento*.

> **Def 15.2 [Navegación restringida]** En la navegación restringida, el usuario tiene prohibido el acceso a determinados ítems si no cumple una serie de condiciones necesarias. Éstas son establecidas por el autor y dependen de cada modo y estructura de aprendizaje concreta.

En ambos modos, navegación por relación conceptual y por conocimiento, la unidad de navegación es el ítem. Sin embargo, las restricciones impuestas para su acceso son distintas. En el primer modo, navegación por relación conceptual, la accesibilidad de un ítem está marcada por el cumplimiento de las reglas de orden, Ro^k, y navegabilidad, $RTnb^k$, existentes dentro de la EC_A^j actual. En el segundo modo, navegación por conocimiento, el acceso e idoneidad de un ítem viene determinado por lo establecido en las reglas de conocimiento, $Rk(EC_A^j)$.

Podemos decir que la navegación por relación conceptual obliga a visitar los ítems en un orden parcial establecido en función de las relaciones conceptuales existentes. Y que la navegación por conocimiento dirige el recorrido del usuario, impidiéndole la visita a los ítems para cuyo contenido no está capacitado y desaconsejándole otros que no le van a proporcionar conocimiento adicional.

4.2 Respecto a la estructura de navegación

En todos los casos, la estructura proporcionada para navegar es una red semántica. Excepto en la navegación por conceptos, ésta coincide con la estructura conceptual de presentación, EC_P^i, incluida en la estructura de aprendizaje que ha sido elegida para el usuario, EC_A^j.

En el caso de la **navegación por conceptos**, la red semántica no explicita el dominio de información. Es decir, la estructura de navegación muestra únicamente los conceptos y relaciones conceptuales incluidas en la estructura conceptual de presentación EC_P^i. Este tipo de navegación es muy útil para obtener una visión de conjunto del sistema hipermedia, por lo que también puede aplicarse a la estructura conceptual de memorización completa, EC_M[1].

En la navegación por conceptos, la red semántica no muestra ítems, ni asociaciones funcionales, por lo que la unidad de navegación es el *concepto*. Cuando el usuario selecciona un concepto obtiene un resumen de la información relativa a éste que existe en el sistema. Dicho resumen se compone con el contenido de los ítems asociados al concepto, de acuerdo a una estructura determinada.

En la **navegación tradicional, por relación conceptual y por conocimiento**, la unidad de navegación es el *ítem*. Es decir, el usuario accede a la información del sistema hipermedia a través de la selección de ítems. Cuando un usuario selecciona un ítem en la red semántica inmediatamente obtiene la información contenida en éste, siempre y cuando sea accesible.

4.3 Respecto al modelo de usuario

La gestión del modelo de usuario es una tarea que el sistema realiza siempre que el usuario interacciona con él, sea cual sea el modo de navegación que emplee. Concretamente la **actualización del estado de conocimiento** del usuario es un proceso que se ejecuta de modo **uniforme** en los cuatro tipos de navegación.

Suponiendo EC_A^j la estructura elegida para el usuario y, por lo tanto, fijado el conjunto de reglas de conocimiento y actualización, $Rk(EC_A^j)$ y $Ru(EC_A^j)$. Siempre que el usuario inspecciona el contenido de un ítem i_j, satisfaciendo el antecedente de accesibilidad de alguna de sus reglas de conocimiento $Rk^r(i_j)$, el sistema ejecuta de forma automática la regla de actualización asociada, esto es $Ru(i_j)$.

En la figura 5 se muestra como, cada vez que el usuario visita un ítem, el sistema evalúa las reglas de conocimiento asociadas para comprobar si con el estado de conocimiento del usuario dicha visita está “permitida” o incluso “aconsejada” de acuerdo a las restricciones de conocimiento establecidas por el autor (véase el capítulo 8, Reglas de conocimiento).

En caso afirmativo, el sistema ejecuta la regla de actualización del ítem visitado, actualizando el grado de conocimiento del usuario respecto a ése y otros ítems. En caso de que la visita esté “prohibida” en todas las reglas de conocimiento evaluadas, la visita del ítem no incrementa el estado de conocimiento del usuario, puesto que no estaba preparado para asimilar correctamente su contenido. Obviamente, si el ítem visitado no tiene ninguna regla de conocimiento asociada, la visita al mismo está siempre “aconsejada” y, por lo tanto, tras su visita siempre se ejecuta su regla de actualización.

[1] Puesto que todas las estructuras de aprendizaje EC_A^j incluyen la EC_M, esta posibilidad es válida cualquiera que se elija.

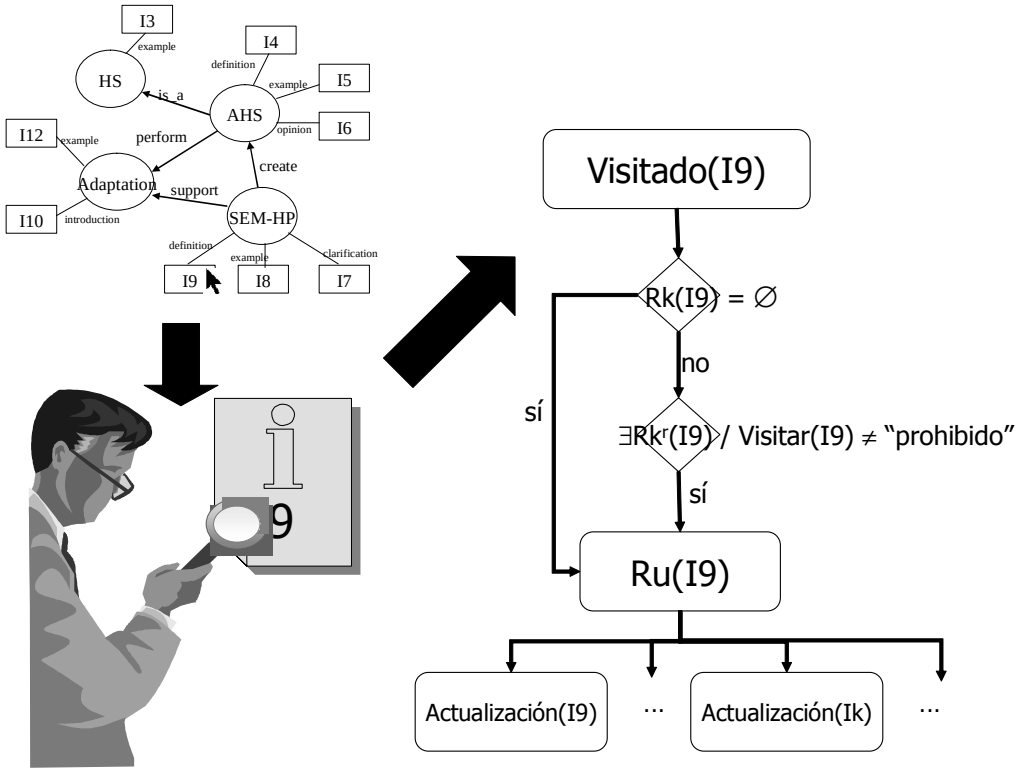


Figura 5: Actualización del estado de conocimiento

En el modo de navegación por conocimiento, el usuario únicamente puede acceder a un ítem si previamente satisface las condiciones de accesibilidad necesarias. Por ello, la visita a un ítem siempre implica la ejecución de su regla de actualización. Esto es, el sistema no tiene que reevaluar las reglas de conocimiento después de la visita a un ítem, porque ya las ha evaluado antes con el fin de determinar si ésta era o no posible.

Observe las ecuaciones 1 y 2 que definen la actualización del estado de conocimiento del usuario tras una visita a i_j durante la navegación por conocimiento (1) y en el resto de modos de navegación (2). Respecto a la ecuación 2 la única particularidad es que en el modo de navegación por conceptos, los ítems visitados son todos aquellos i_1, i_2,..., i_t cuyo contenido se muestra en el resumen del concepto seleccionado, por ello j = 1. .t.

$$\text{Visitado}(i_j) \rightarrow \text{Ru}(i_j) \qquad (1)$$

$$\text{Visitado}(i_j) \text{ and } [\text{Rk}(i_j)=\varnothing \text{ or } \exists \text{Rk}^r(i_j) / \text{Visitar}(i_j) \neq \text{“prohibido”}] \rightarrow \text{Ru}(i_j) \qquad (2)$$

Asimismo, durante todos los modos de navegación, el sistema contabiliza automáticamente el **número de visitas** realizadas por el usuario. Para ello, cada vez que el usuario accede al contenido de un ítem i_j, de forma autónoma o dentro del resumen de un concepto, se incrementa en uno el atributo visitas(i_j) en su modelo de usuario. Obviamente las visitas de un concepto, visitas(c_k), sólo se actualizan durante la navegación por conceptos, cada vez que el usuario inspecciona el resumen generado para éste.

También, la **actualización explícita** del modelo es común a todos los tipos de navegación, es decir, el usuario puede modificar el contenido de su modelo

independientemente del tipo de navegación que utiliza. Sin embargo, cabe esperar que la actualización de determinada información sea más frecuente en un modo que en otro.

Por ejemplo, durante la navegación por conceptos el usuario se preocupa especialmente por personalizar la estructura con la que desea que se generen los resúmenes. Mientras que fijar una meta de conocimiento, establecer sus preferencias o indicar intereses es una tarea que habitualmente el usuario realiza durante la navegación por conocimiento.

4.4 Respecto a la adaptación realizada

La adaptación realizada depende, por supuesto del usuario, pero también del modo que éste utiliza para recorrer la información. Esto es, *en cada modo de navegación se realiza un proceso de adaptación propio y diferente*. Dicho proceso de adaptación depende del modelo de usuario, de la estructura y unidad de navegación, y de las restricciones empleadas.

En todos los modos de navegación se incluye directamente sobre la estructura de navegación una serie de **información** con objeto de asistir al usuario en su recorrido. El tipo de dicha información depende del modo de navegación utilizado. Es decir, en la navegación por conceptos se indica para cada concepto el número de visitas realizadas a éste, *visitas(c_k)*, mientras que en la navegación por relación conceptual lo que se anota es el número de veces que cada ítem ha sido visitado por el usuario, *visitas(i_j)*. Del mismo modo, en la navegación por conocimiento se específica junto a cada ítem y concepto, el grado de conocimiento que posee actualmente el usuario, *K(i_j) y K(c_k)* respectivamente, mientras que en la navegación tradicional simplemente se notifica *si el ítem ha sido o no visitado* con anterioridad.

En los modos de navegación restringida se establece sobre la propia estructura de navegación qué ítems no puede visitar el usuario. Por ejemplo, en la navegación por conocimiento, esta tarea se hace **ocultando los ítems inaccesibles**, de forma que el usuario comprenda que dado su estado de conocimiento actual y según las reglas de conocimiento presentes, $Rk(EC_A^j)$, esos ítems no están disponibles.

También en los modos de navegación restringida, ya sea por relación conceptual o por conocimiento, se **deshabilitan los ítems no accesibles**, de acuerdo a las reglas de orden y navegabilidad en el primero (Ro^k, $RTnb^k$) o según las reglas de conocimiento en el segundo (Rk^j). Esto impide que el usuario tenga acceso al contenido de un ítem de información para el que no satisface las condiciones exigidas, aún cuando se empeñe en seleccionarlo.

Asimismo, los ítems que es aconsejable que el usuario visite son anotados de forma positiva en ambos modos de navegación. Concretamente en la navegación por conocimiento, esta **anotación positiva** anima al usuario a seleccionar los ítems para los que se satisfacen las restricciones de accesibilidad e idoneidad impuestas en las reglas de conocimiento. Desaconsejando al usuario visitar los ítems, que aún siendo accesibles, no van a proporcionarle, según el autor, conocimiento relevante.

Además en la navegación por conocimiento se anotan de forma especial aquellos ítems que al ser visitados acercan al usuario a la consecución de sus *intereses* o *meta de conocimiento*. Y el usuario tiene la posibilidad de solicitar una **ruta guiada**, donde en

cada paso de navegación se le indica de forma precisa el ítem que debe visitar a continuación hasta alcanzar el estado de conocimiento deseado.

Por su parte, *el comportamiento del usuario durante la navegación libre es analizado* para determinar las estrategias de navegación seguidas mayoritariamente por los usuarios del sistema, extraer de éstas las estructuras mentales que utilizan, y compararlas con las estructuras de navegación definidas por el autor. Tras este análisis, el sistema proporciona al autor las diferencias significativas entre las estructuras reales y las estructuras virtuales que los usuarios tienen en su cabeza. Después, el autor puede modificar las estructuras de navegación, permitiendo que los futuros individuos que usen el sistema se beneficien del conocimiento y experiencia proporcionados implícitamente por un grupo de usuarios del mismo. Se trata pues de una **adaptación al grupo** que beneficia de forma particular cada uno de los miembros (capítulo 21, Retroalimentación adaptativa).

En la tabla 2 se resumen los métodos de adaptación ejecutados durante cada modo de navegación. La navegación tradicional se describe en la sección 5 del presente capítulo, mientras que cada uno de los otros modos es desarrollado en un capítulo aparte: Navegación por Conceptos en el capítulo 16, Navegación por Relación conceptual en capítulo el 17 y Navegación por Conocimiento en capítulo el 18.

Tabla 2: Adaptación en los modos de navegación

Elección de la EC_A en función del perfil del usuario			
	Navegación Tradicional	Anotación de los ítems visitados.	Retroalimentación adaptativa
	Navegación por Conceptos	Anotación del número de visitas a cada concepto.	
		Personalización de la estructura de los resúmenes.	
	Navegación por Relación conceptual	Anotación del número de visitas a cada ítem.	Ocultación y deshabilitación de ítems no accesibles
		Anotación positiva de conceptos disponibles e ítems accesibles.	
	Navegación por Conocimiento	Anotación del grado de conocimiento sobre ítems y conceptos.	
		Anotación negativa de los ítems accesibles pero no idóneos.	
		Anotación positiva de ítems deseables para alcanzar los intereses del usuario.	
		Generación de rutas guiadas a la meta.	

5. NAVEGACIÓN TRADICIONAL

Aunque los beneficios de la navegación guiada son indudables, también es cierto que los usuarios están acostumbrados a una forma de navegación donde no existen, salvo por cuestiones de seguridad, restricciones para acceder a los hiperdocumentos. Por este motivo, consideramos que el sistema debe proporcionar un modo de navegación que permita al usuario recorrer la información tal y como están habituados a hacerlo en los

sistemas web no adaptativos. Este tipo de navegación es denominado navegación tradicional y puede definirse como sigue:

> **Def 15.3 [Navegación tradicional]** La navegación tradicional es un modo de navegación libre, donde la red semántica proporcionada para navegar coincide con la estructura conceptual de presentación EC_P^i incluida en la estructura de aprendizaje elegida para el usuario, EC_A^j. Permite acceder al contenido de cualquier ítem sin más que seleccionarlo en la red. No tiene pues restricciones de acceso para acceder a la información y la unidad de navegación es el ítem.

Aunque este tipo de navegación se corresponde con la navegación web tradicional, presenta sobre ésta una serie de ventajas importantes, derivadas en gran parte de la utilización de una red semántica como estructura de navegación (véase la sección 1.2) y de que ésta haya sido elegida de forma específica para el perfil del usuario (capítulo 14, Elección de la EC_A).

Otra ventaja importante, es que la navegación se realiza sobre una presentación y no sobre la estructura conceptual de memorización completa. Las estructuras conceptuales de presentación, EC_P^i, muestran un subconjunto del dominio de conocimiento representado en la EC_M, lo que las convierte en una excelente estructura de navegación desde el punto de vista de la orientación. En este sentido podemos destacar dos de sus características:

- Al tratarse de una vista parcial, normalmente una EC_P^i tiene una **extensión reducida**. Esta característica la hace muy adecuada para ser navegada, ya que los problemas de desorientación se reducen considerablemente cuando el proceso de navegación se realiza sobre una estructura de un tamaño razonable.

- Por otro lado, una EC_P^i **se centra en una parcela de conocimiento**, de manera que todos sus conceptos e ítems giran en torno a un mismo tema o varios temas muy relacionados. Por lo tanto, es más difícil que un usuario se sienta perdido cuando recorre dicha estructura y accede a la información que en ésta se le proporciona.

Supongamos el ejemplo de la figura 6, donde se muestra una estructura conceptual de memorización y una presentación EC_P^i creada a partir de ella.

Observe que al reducir el número de ítems, conceptos y relaciones mostradas en EC_P^i, es posible obtener una visión de conjunto de la presentación, de forma rápida y sin apenas esfuerzo.

Se ha excluido de la presentación la información relacionada con los sistemas software en general y su evolución. Esto es, los conceptos *Software System* y *Evolution*, y los ítems I1, I15 e I14 asociados a éstos. De esta forma, la presentación se centra únicamente en los sistemas hipermedia adaptativos y el modelo SEM-HP. Lo cual permite evitar que el usuario se distraiga con conceptos vecinos que actualmente no le deben preocupar.

También se han dejado fuera de la presentación los ítems I10 e I2, que a pesar de tratar sobre los conceptos *HS* y *AHS* se han considerado no adecuados para dicha presentación.

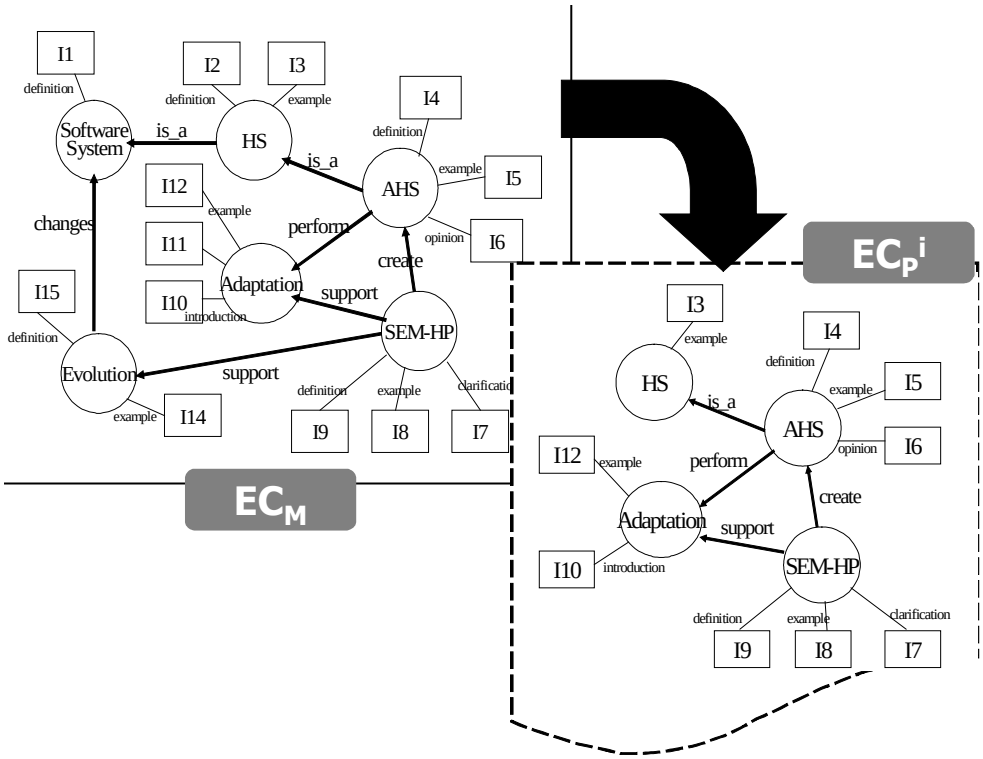


Figura 6: Creación de una presentación, EC_P^i

Suponiendo que la estructura de aprendizaje seleccionada se define sobre EC_P^i, esto es, $EC_A^j = (EC_M, Rw, MU, EC_P^i, RTnb^k, Ro^k, Ru^j, Rk^j)$, el usuario realiza su navegación utilizando un *browser* similar al que se muestra en la figura 7.

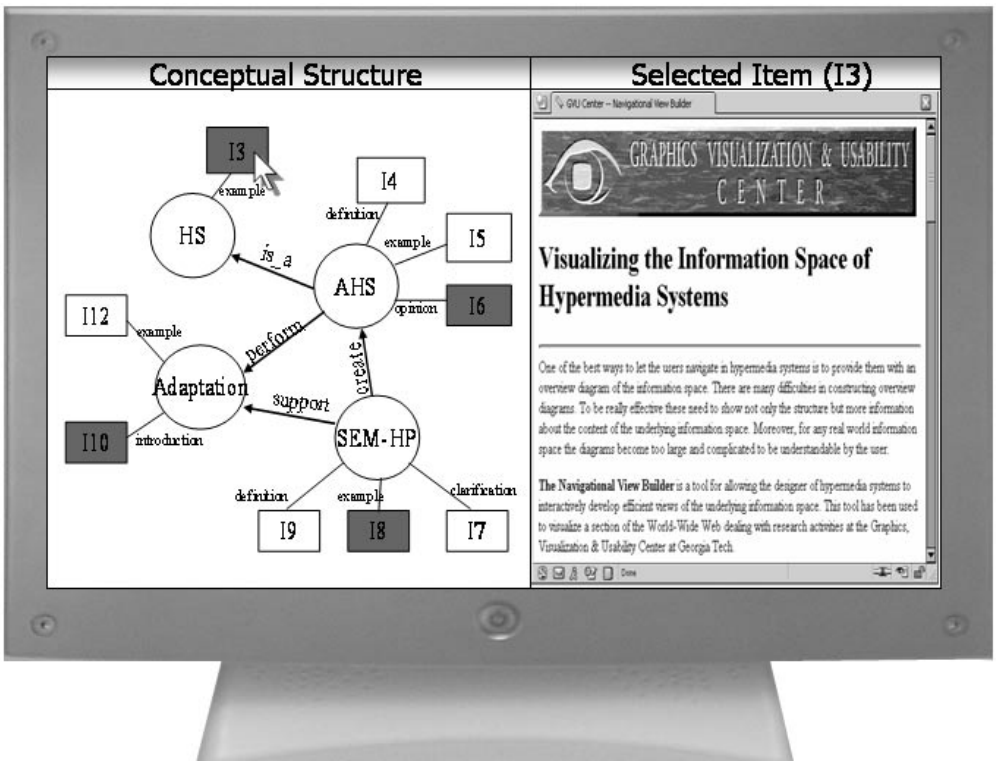


Figura 7: Interfaz para navegar en modo tradicional sobre EC_A^j

La interacción es muy sencilla, para seleccionar un ítem (I3), el usuario hace doble clic sobre el rectángulo que lo representa en la red semántica de la derecha (EC_P^i), y automáticamente obtiene el contenido del mismo a su izquierda.

Como puede observarse en la figura 7, en este modo de navegación, la interfaz facilitada al usuario tiene únicamente dos marcos. Uno donde se muestra la estructura de navegación, en este caso la red semántica que representa la EC_P^i incluida en la estructura de aprendizaje elegida. Y otro que visualiza el contenido del último ítem seleccionado. En la figura, los dos marcos se organizan en mosaico vertical, a la izquierda la estructura de navegación y a la derecha el ítem actualmente seleccionado.

La única adaptación realizada sobre la estructura de navegación es la anotación mediante un color especial de aquellos ítems que han sido visitados previamente por el usuario. Para ello, el sistema consulta las entradas del modelo de usuario asociadas a los ítems incluidos en la presentación actual. Y, para cada ítem $i_j \in I(EC_P^i)$ tal que visitas(i_j)≥1, rellena en morado el interior del rectángulo que representa el ítem en la red semántica. Este es el caso de los ítems I3, I6, I8 e I10 en el ejemplo de la figura 7.

CAPÍTULO 16

Navegación por Conceptos

Resumen

En este capítulo se justifica la necesidad de un modo de navegación libre basado en conceptos, especialmente adecuado para recorrer el dominio de conocimiento completo, esto es la estructura conceptual de memorización. Se establece para ello una estructura de navegación conceptual, donde el dominio de información está semi-presente gracias a la generación de resúmenes de conceptos. Se define una estructura de composición genérica y un mecanismo dinámico que, a partir de ésta y del dominio de información del concepto seleccionado, permite generar su resumen. Se describe una interfaz de navegación adecuada para la navegación por conceptos, que permite al usuario personalizar un resumen, e incluso la estructura de composición genérica de todos los resúmenes. Finalmente se refieren las técnicas de adaptación utilizadas, y la actualización del modelo de usuario, así como la conveniencia de aplicar este modo de navegación, también, en presentaciones extensas.

Tabla de contenidos

Navegación por Conceptos

1. INTRODUCCIÓN

Por poco complejo que sea el tema tratado en un sistema hipermedia, la estructura conceptual de memorización que representa su dominio de conocimiento suele contener un número importante de conceptos y trozos de información relativos a éstos. En este sentido, es sensato reconocer que, en la mayoría de los casos, la extensión de la estructura conceptual de memorización va a ser considerable.

En la fase de presentación, el autor prepara varias estructuras de presentación, EC_P^i, a partir de la estructura conceptual de memorización, EC_M. Cada una de estas presentaciones captura una parte específica del dominio conceptual y de información completo, por lo que tiene un tamaño bastante menor. Después, cada usuario recorre una estructura u otra en función del subdominio de conocimiento que le interesa, y de su experiencia de navegación y en la materia (capítulo 14, Elección de la EC_A).

El uso de la EC_P^i como estructura de navegación permite disminuir, durante la navegación tradicional (capítulo 15), los problemas de desorientación derivados de un excesivo volumen de ítems de información. Es decir, el recorrido libre realizado por el usuario durante la navegación tradicional marcha adecuadamente gracias al reducido tamaño de la estructura de navegación utilizada. En caso contrario, esto es si la estructura fuese muy extensa, el usuario podría perderse dentro de la gran maraña de información.

Sin embargo, no hay que ignorar el hecho de que algunos usuarios pueden estar interesados en visualizar la estructura conceptual de memorización completa, con la intención de conocer desde una perspectiva global la semántica de sus conceptos y las relaciones entre ellos. En este caso la estructura de navegación no se corresponde con una presentación parcial sino con la estructura de memorización entera, que obviamente posee un número mucho mayor de conceptos, ítems y relaciones.

El problema que se plantea es el siguiente: ¿Cómo se puede realizar un proceso de navegación libre sobre la EC_M, y evitar los consecuentes problemas de desorientación? Puesto que se trata de una navegación libre donde no existen restricciones que permitan dirigir o guiar al usuario, la solución pasa necesariamente por reducir el tamaño de la estructura de navegación y se materializa en un modo de navegación basado en conceptos.

La forma en que dicha solución es llevada a cabo se explica detalladamente a lo largo del presente capítulo. Donde la idea general es proporcionar una estructura de navegación centrada en los conceptos, donde inicialmente se ocultan todos los ítems de información, que únicamente se muestran cuando el usuario se interesa por el concepto al que se asocian.

2. ESTRUCTURA DE NAVEGACIÓN POR CONCEPTOS

Entonces, para satisfacer la citada necesidad de navegación, el sistema permite al usuario un tipo de navegación basado en conceptos, donde la estructura de navegación

es una *estructura conceptual de presentación especial que captura el dominio conceptual completo*. Concretamente, el usuario puede navegar a través de una estructura en la que **únicamente se muestran los conceptos y las relaciones conceptuales** de la estructura conceptual de memorización.

En el siguiente ejemplo se muestra la estructura de navegación, figura 1.b, proporcionada en la navegación por conceptos a partir de una estructura conceptual de memorización concreta, figura 1.a.

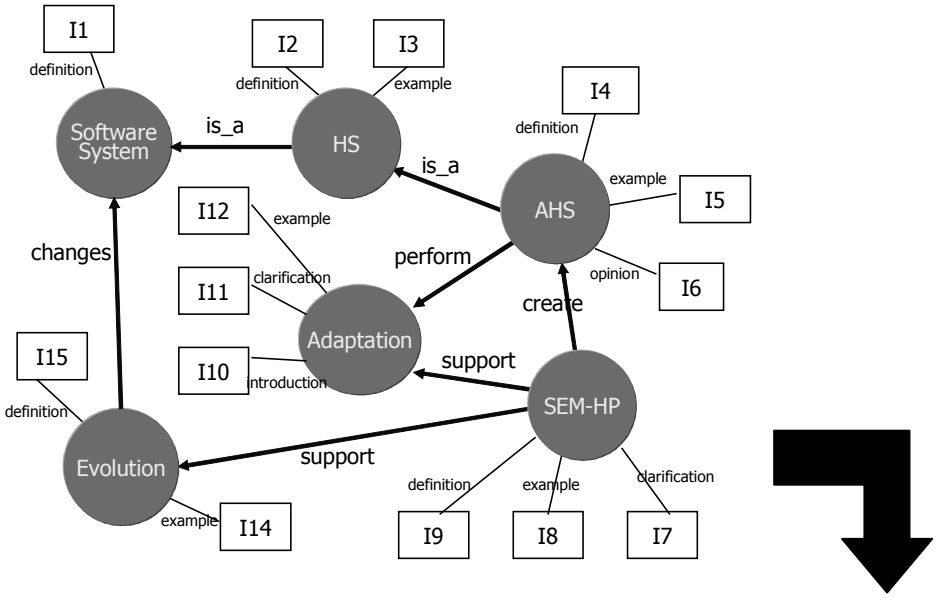


Figura 1.a: Una EC_M

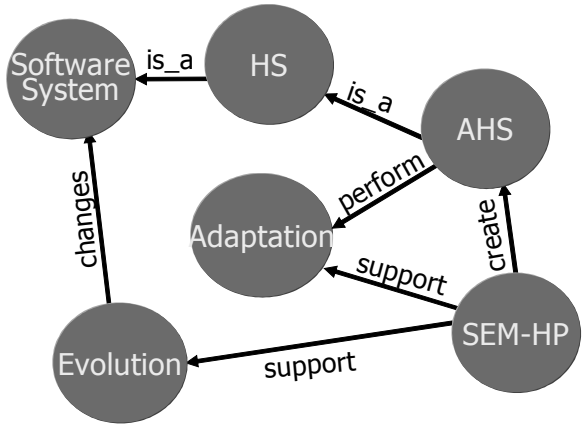


Figura 1.b: EC_P sólo con el dominio conceptual

Observe que al no mostrar el dominio de información, es decir los ítems de información y las asociaciones funcionales que los ligan a los conceptos, la estructura de navegación se simplifica, de forma que el usuario puede visualizar íntegramente y con claridad el dominio conceptual del sistema.

Esta estructura de navegación exclusivamente conceptual tiene un evidente valor pedagógico, ya que permite al usuario adquirir una idea global de los conceptos implicados en el sistema hipermedia y de las relaciones que existen entre ellos. Esta

concepción total del sistema hipermedia puede ser de gran utilidad al usuario en sus futuras navegaciones, estando en su mano acceder a los ítems, no de una forma aleatoria o desordenada, sino en un orden coherente con los conceptos y sus asociaciones.

3. ACCESO A LA INFORMACIÓN

En la sección anterior se ha descrito la estructura navegación utilizada en el modo de navegación por conceptos. Sin embargo, para admitir un proceso de navegación sobre dicha estructura conceptual, es necesario que de alguna manera participe el dominio de información. La cuestión que se plantea ahora es cómo.

Una primera posibilidad es que, al seleccionar un concepto, se desplieguen automáticamente los ítems asociados a éste, de forma que el usuario pueda seleccionar aquellos que desee. En el ejemplo de la figura 2 se muestra el resultado de seleccionar el concepto *AHS* en la estructura de navegación de la figura 1.b. En este momento el usuario puede acceder al contenido de cualquiera de los tres ítems mostrados, esto es I4, I5 e I6.

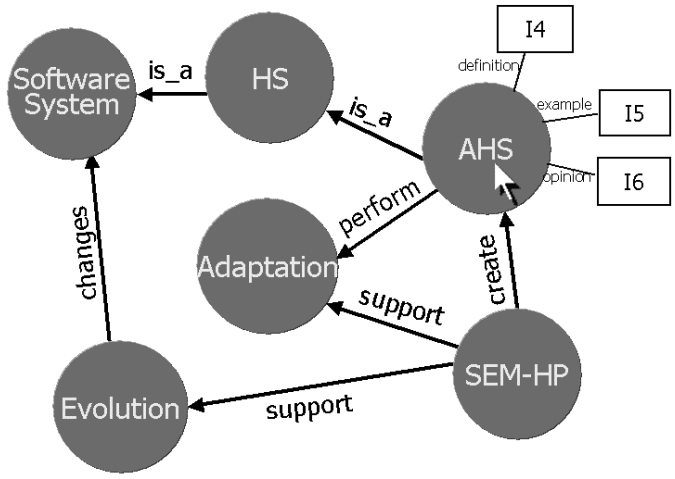


Figura 2: Despliegue del concepto AHS

Esta solución, aunque muy práctica para reducir la extensión de la estructura de navegación, hace que se pierda el carácter esencialmente conceptual de este modo de navegación. Esto es, en la navegación por conceptos el usuario está preocupado de aprender conceptos absolutos en lugar de trozos de información, por lo que no tiene mucho sentido que finalmente se vea obligado a seleccionar ítems concretos.

La solución anterior se descarta para la navegación por conceptos, no obstante puede utilizarse para reducir el tamaño de las presentaciones que así lo requieran en el resto de modos de navegación. En su lugar, cuando el usuario selecciona un concepto va a obtener un documento que integra múltiple información acerca de éste. Por lo tanto, el usuario no trabaja con un conjunto de ítems sueltos que tratan el concepto seleccionado, sino con un **resumen** del mismo, que organiza la información contenida en algunos de esos ítems.

De esta forma, cada vez que el usuario selecciona un concepto, está haciendo presente parte del dominio de información que estaba oculto, concretamente la parte relacionada

con el concepto elegido. Además, la información es explicitada de forma conexa y no a través de piezas sueltas, lo cuál confiere cierta coherencia al modo de navegación conceptual.

> **Def 16.1 [Navegación por conceptos]** La navegación por conceptos es un modo de navegación libre, donde la estructura proporcionada para navegar muestra únicamente conceptos y relaciones conceptuales. La unidad de navegación es el concepto, de modo que al seleccionar un concepto, el usuario obtiene un resumen compuesto con varios ítems de información asociados a éste.

En las siguientes secciones se explica en qué consiste el resumen de un concepto, quién especifica su estructura y cuándo se genera.

4. RESUMEN DE UN CONCEPTO

El resumen de un concepto no es un ítem más asociado al concepto y tampoco exige, por parte del autor, ningún esfuerzo adicional para su creación, salvo la especificación de una estructura estándar. Es más, el resumen de un concepto no tiene que existir de forma previa a la solicitud del usuario, sino que se genera en ese mismo instante mediante la composición de los ítems asociados al concepto.

4.1 Estructura de composición genérica

Una vez determinado el material con el que se va a construir el resumen de un concepto, esto es, los ítems ligados funcionalmente al concepto, el siguiente paso es establecer como se va a componer e integrar toda esa información.

> **Def 16.2 [Estructura de composición]** La forma en que se reúnen en un único "documento" el conjunto de ítems que integran el dominio de información asociado directamente a un concepto constituye la estructura de composición del resumen generado para éste.

La estructura de composición utilizada para crear los resúmenes obedece a la perspectiva lingüística sistémico funcional propuesta por Halliday [Halliday, 85] [Halliday, 85b]. Según esta teoría cada texto tiene una estructura que puede ser expresada como un conjunto de elementos (a) y el orden en que preferentemente éstos aparecerán en el texto (b). Dentro de los elementos que constituyen el texto algunos son opcionales y otros de obligada presencia (c).

Además, de acuerdo a la teoría seguida, los textos de un mismo género se caracterizan por presentar una estructura genérica potencial común, denominada de forma abreviada GSP (*Generic Structure Potencial*). En nuestro caso, la estructura de composición de los resúmenes puede ser vista como una GSP, donde se especifican los tres aspectos indicados arriba:

a) Los elementos que forman el texto, en este caso el resumen de un concepto, son los ítems asociados al concepto mediante alguna relación funcional. Denominamos **elementos-resumen(c_k)** al conjunto de ítems implicados en el resumen del concepto c_k. Dicho conjunto se constituye tal y como se expresa en la ecuación 1.

$i_j \in$ elementos-resumen(c_k) si $\exists\ a_f \in$ Af(EC_M) tal que $a_f = <c_k, r_f, i_j>$ (1)

De acuerdo a lo especificado en (1), los elementos del resumen creado para el concepto *AHS* de la figura 1.b, son los trozos de información identificados como I4, I5 e I6. Por lo que, elementos-resumen(AHS) = {I4, I5, I6}.

b) El **orden que ocupa cada elemento**, en este caso ítem, en un resumen esta determinado por su rol. Es decir, la relación funcional r_f que liga el ítem i_j al concepto c_k es, a la hora de confeccionar el resumen de c_k, el indicador que especifica la posición que ocupará dentro del resumen la información contenida en el ítem i_j.

c) También el **carácter obligatorio/opcional** de un ítem en el resumen está determinado por su rol. De manera que la inclusión inicial de un ítem i_j en el resumen de un concepto c_k depende nuevamente de la relación funcional r_f que lo liga a dicho concepto.

Los elementos concretos que componen un resumen **(a)** varían de un concepto a otro y son obtenidos automáticamente por el sistema a partir de las asociaciones funcionales definidas previamente por el autor en la EC_M. Por lo tanto, para especificar completamente la estructura de composición de los resúmenes, el autor tan sólo debe precisar los aspectos b) y c) indicados anteriormente.

En **b)** el autor tiene que establecer un *orden estricto para el conjunto de roles existentes*, esto es, no puede dejar un rol sin orden asociado y no puede asignar a dos roles diferentes el mismo orden.

El orden de un rol coincide con la posición en que éste aparecerá en los resúmenes. De modo que dados dos roles, r_f^1 y r_f^2, si el orden dado a r_f^2 es mayor que el de r_f^1, orden(r_f^1) < orden(r_f^2), quiere decir que r_f^1 va a preceder a r_f^2 en los resúmenes de los conceptos.

Por defecto, el conjunto de roles con que un ítem puede ligarse a un concepto en la estructura conceptual de memorización es enumerado en la tabla 1. No obstante, el autor puede añadir nuevos roles si los necesitase.

Tabla 1: Roles o relaciones funcionales

Rf	
introducción	recomendación
definición	explicación
ejemplo	análisis
aclaración	ampliación
bibliografía	justificación
opinión	enfoque
comparación	valoración
antecedentes	

En **c)** el autor tiene que especificar para cada rol si su carácter es **"obligatorio" u "opcional"** en el resumen. Por defecto todos los roles son "obligatorios", por lo que el

resumen de un concepto mostraría el contenido de todos los ítems asociados a éste, cualquiera que sea su rol.

Cuando el autor convierte un rol r_f^i en "opcional", hace que los ítems i_j asociados mediante esa función r_f^i a un concepto c_k no aparezcan inicialmente en su resumen. El autor sólo debe asignar carácter "opcional" a un rol, si considera que la información de los ítems que tratan un concepto con ese rol o función no es imprescindible para entender el concepto.

Las decisiones tomadas en b) y c) son representadas internamente mediante un conjunto ordenado de parejas al cuál denominamos **organización-resumen(EC_M)**. Cada pareja incluida en dicho conjunto es expresada como (r_f^i, $cter^i$), y corresponde a un rol r_f^i distinto (véase la ecuación 2).

$$\neg\exists\ [(r_f^i, cter^i) \in \text{organización-resumen}(EC_M) \wedge (r_f^k, cter^k) \in \text{organización-resumen}(EC_M)]$$
$$\text{tal que: } i \neq k \text{ y } r_f^k = r_f^i \qquad (2)$$

Obviamente, el número total de parejas incluidas en el conjunto organización-resumen(EC_M) coincide con el total de roles diferentes existentes en la EC_M.

La posición en el conjunto organización-resumen(EC_M) de la pareja (r_f^i, $cter^i$) define el orden que corresponderá al rol r_f^i en el resumen de un concepto (véase la ecuación 3).

$$\text{Dado organización-resumen}(EC_M) = \{(r_f^1, cter^1), (r_f^2, cter^2), ..., (r_f^n, cter^n)\},$$
$$\text{se cumple que: } \forall_{i=1..(n-1)},\ \text{orden}(r_f^i) < \text{orden}(r_f^{i+1}) \qquad (3)$$

El segundo elemento $cter^i$ de cada pareja (r_f^i, $cter^i$), indica el carácter opcional u obligatorio del rol asociado, esto es r_f^i. Por lo tanto, únicamente puede tomar los valores "opcional" u "obligatorio".

De este modo, el resumen de cada concepto $c_k \in C(EC_M)$ queda perfectamente determinado por la combinación de: su dominio de información asociado, *elementos-resumen(c_k)*, y la estructura de composición genérica, *organización-resumen(EC_M)*.

4.2 Creación del resumen

A partir de la estructura de composición especificada, el sistema construye el resumen de un concepto c_k, una vez que éste ha sido solicitado, de la siguiente manera:

- En todo resumen existe una sección para cada rol, la cuál se encabeza o titula con el nombre de dicho rol, esto es r_f^i. De manera que el resumen se divide en tantas partes como roles diferentes existen en la EC_M.

- La sección asociada a cada rol aparece en el resumen en la posición que se especifica en la estructura de composición. Concretamente, dada la organización-resumen(EC_M) = {(r_f^1, $cter^1$), (r_f^2, $cter^2$), ..., (r_f^n, $cter^n$)}, la sección correspondiente al rol r_f^i es la i-esima sección del resumen.

- Si en la EC_M el concepto c_k no presenta ningún ítem asociado funcionalmente mediante el rol r_f^i (ecuación 4), en su resumen la sección del rol r_f^i aparece vacía y sombreada en gris. Por cuestiones de espacio también podría eliminarse.

$$\neg\exists i_j \in \text{elementos-resumen}(c_k) \text{ tal que: } <c_k, r_f^i, i_j> \in Af(EC_M) \qquad (4)$$

- Si el concepto c_k tiene en la EC_M uno o varios ítems asociados funcionalmente mediante el rol r_f^i, la sección correspondiente en su resumen muestra para cada uno de estos ítems ($\forall i_j \in$ elementos-resumen(c_k) tal que $<c_k, r_f^i, i_j> \in Af(EC_M)$):

    a) el identificador y el contenido del ítem i_j si el rol r_f^i es obligatorio, esto es $cter^i$ = "obligatorio" en organización-resumen(EC_M), o

    b) únicamente el identificador del ítem i_j si r_f^i es opcional, esto es $cter^i$ = "opcional" en organización-resumen(EC_M).

La figura 3 muestra una posible estructura de composición, de acuerdo a la cual, la primera sección de un resumen visualiza los ítems asociados al concepto con el rol "introducción", siguiéndole a ésta las secciones, también obligatorias, "definición" y "ejemplo", tras las que aparecen dos secciones opcionales "opinión" y "bibliografía", etc.

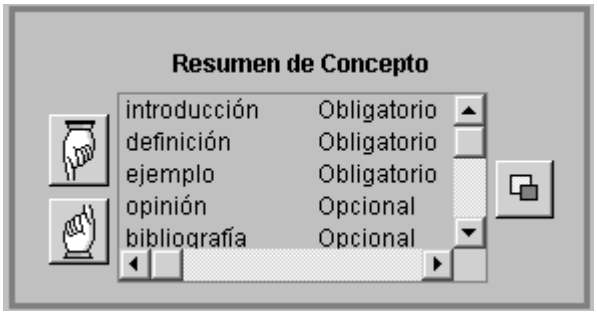


Figura 3: Una estructura de composición de resúmenes

La organización del resumen especificada en la interfaz de la figura 3 se formaliza como sigue: organización-resumen(EC_M) = {("introducción", "obligatorio"), ("definición", "obligatorio"), ("ejemplo", "obligatorio"), ("opinión", "opcional"), ("bibliografía", "opcional"), ...}.

En la figura 4 se muestra el resumen generado para el concepto *AHS* de la figura 1.b. El resumen se obtiene instanciando la organización genérica que acabamos de indicar, organización-resumen(EC_M), con el conjunto de ítems específicos del concepto *AHS*, elementos-resumen(AHS).

En dicho resumen vemos cómo las secciones se colocan el orden especificado: 1º Introduction, 2º Definition, 3º Example, 4º Opinión, etc. Puesto que en el conjunto de elementos-resumen(AHS) no existe ningún ítem asociado al concepto *AHS* mediante el rol "introducción", la primera sección de su resumen aparece vacía y sombreada en gris.

La segunda sección muestra el contenido del ítem I4 asociado con el rol "definition" al concepto resumido. Por su parte la tercera sección, correspondiente al rol "example", muestra el contenido del ítem I5. La última sección no visualiza el contenido de ningún ítem, pero en este caso no se debe a la inexistencia de ítems con el rol "opinión", sino a que dicho rol tiene carácter "opcional" en la organización-resumen(EC_M).

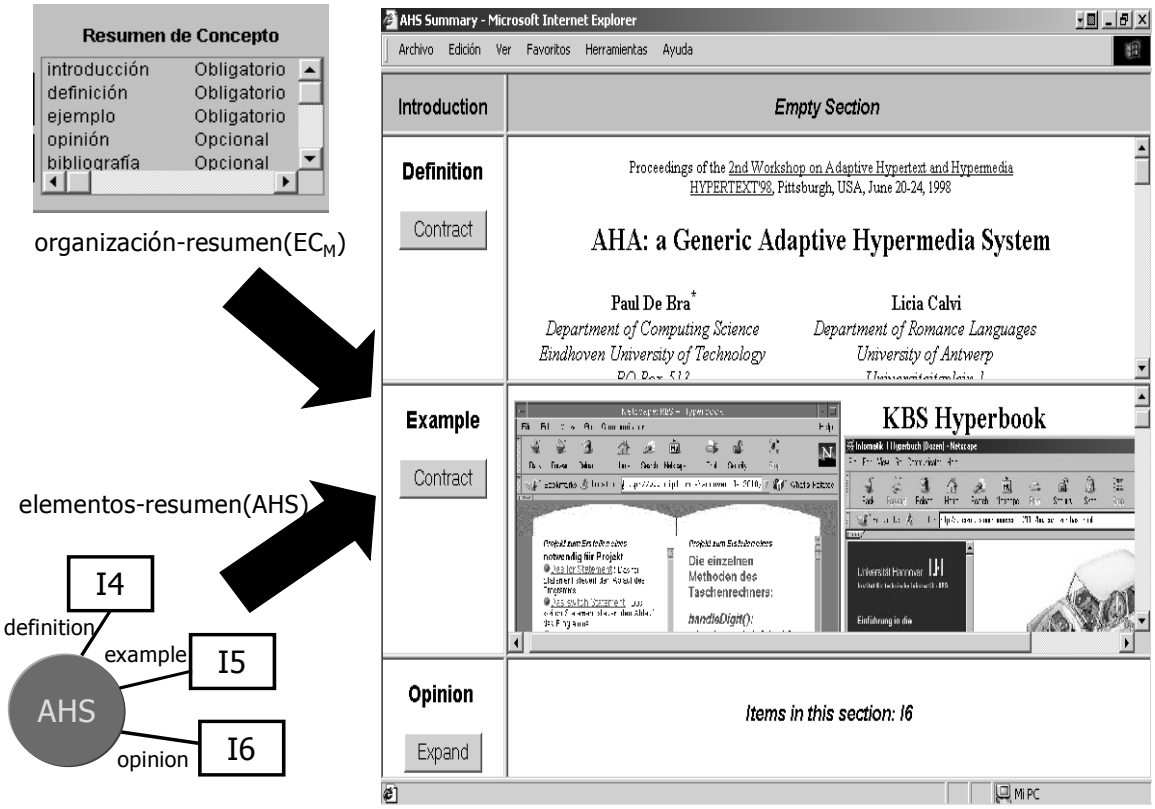


Figura 4: Resumen para el concepto AHS

Es fácil distinguir las secciones opcionales de las secciones vacías, porque las primeras tienen fondo blanco (en lugar de gris), informan de la existencia de uno o varios ítems con el rol considerado, y proporcionan un botón para poder visualizarlos (*Expand*).

El proceso de generación de los resúmenes es una tarea que el sistema realiza de forma automática cada vez que un usuario, utilizando el modo de navegación por conceptos, selecciona un concepto. Esto es, los resúmenes de los conceptos no tienen por qué existir de forma previa a su uso, ya que el sistema tiene la capacidad de generarlos dinámicamente en el momento que se solicitan.

Para mejorar el tiempo de respuesta, es posible crear y almacenar de antemano el resumen de cada concepto de acuerdo a la estructura de composición estándar definida por el autor, y al dominio conceptual y de información representado en la EC_M. Estos resúmenes estarían inmediatamente disponibles para todos los usuarios que los requiriesen. Sin embargo, si el autor realiza determinadas modificaciones sería necesario generarlos de nuevo. Dependiendo de la modificación podrían verse alterados uno, varios o todos los resúmenes.

Atendiendo a su alcance podemos distinguir dos tipos de cambios que afectan a los resúmenes: modificaciones en los elementos de la EC_M y cambios en la organización estándar del resumen.

Los *cambios en la organización-resumen(EC_M)* afectan a todos los resúmenes generados. Por ejemplo, si se cambia el orden de los roles hay que cambiar el orden de las secciones en todos los resúmenes. Del mismo modo, si se cambia el carácter de un

rol (opcional ⇔ obligatorio) es necesario mostrar u ocultar en cada resumen el contenido de los ítems asociados con ese rol al concepto resumido.

Por el contrario, los *cambios en los elementos de la EC_M* afectan únicamente a un grupo limitado de resúmenes. En general, los cambios en el Sistema de Memorización que repercuten en los resúmenes son aquellos que, para algún concepto c_k, implican añadir o borrar ítems de su conjunto elementos-resumen(c_k). Este tipo de cambios requiere ampliar o reducir el contenido de algunas secciones en los resúmenes afectados.

Concretamente, si el autor elimina o añade una asociación funcional $<c_k, r_f^i, i_j>$ en la EC_M sólo se ve afectado el resumen del concepto implicado, esto es c_k. Si elimina un ítem i_j deben reconsiderarse los resúmenes de todos los conceptos c_k al que éste estaba asociado (esto es: $\exists <c_k, r_f, i_j> \in Af(EC_M)$). Por supuesto, si se elimina un concepto en la EC_M hay que suprimir su resumen, del mismo modo que si se añade un nuevo concepto hay que preparar un resumen inicial.

5. Estructura de Composición Personalizada

5.1 Personalizar la organización de los resúmenes

La estructura de composición de los resúmenes no es algo que permanezca oculto para el usuario, ya que éste puede percibirla de dos formas diferentes:

a) directamente estudiando en la interfaz de su modelo de usuario la lista ordenada de roles (figura 3), o

b) indirectamente a través de un resumen generado, al observar el orden y contenido de sus diferentes secciones (figura 4).

Asimismo, aunque la estructura de composición de los resúmenes es inicialmente definida por el autor de una forma estándar para todos los usuarios del sistema, lo ideal es que cada uno pueda personalizarla. Esta posibilidad, permite ajustar, aún más, el tipo de navegación por conceptos a las preferencias y gustos individuales de cada usuario.

La adaptación realizada se limita a la organización del resumen, de forma que el usuario puede decidir el orden y carácter de los roles en los resúmenes, aunque no el contenido de los mismos. Obviamente, el usuario no puede modificar la estructura conceptual de memorización, ni por lo tanto, los elementos que constituyen el resumen de un concepto, pero sí la forma en que éstos se muestran y organizan.

Para llevar a cabo el proceso de **personalización de los resúmenes**, el usuario tiene de nuevo dos alternativas, estrechamente relacionadas con la forma de percibir la estructura de composición:

a) modificar directamente la estructura de composición en el modelo de usuario, usando las funciones que se facilitan en el mismo para reordenar los roles y modificar el carácter “obligatorio” u “optativo” de cada rol (ver capítulo 12, Modelo de usuario), o

b) modificar indirectamente la estructura de composición general de los resúmenes cambiando la estructura de un resumen concreto.

Esta segunda alternativa es llevada a cabo por el usuario mientras que lee el resumen de un concepto, de una forma tan sencilla y cómoda como la siguiente:

- Para cambiar el carácter de un rol, el usuario debe utilizar los botones *Expandir* / *Contraer* que aparecen en la correspondiente sección del resumen. Una sección optativa se expande al pulsar el botón *Expandir*, del mismo modo que una sección obligatoria se contrae con el botón *Contraer*. En el primer caso, la sección expandida pasa a visualizar el contenido de los ítems que antes tan sólo enumeraba. En el segundo caso, la sección contraída, deja de mostrar el contenido de los ítems, presentando únicamente sus identificadores.

- Para cambiar el orden de los roles, el usuario puede reordenar las distintas secciones del resumen, reubicándolas a su antojo sin más que arrastrar cada sección desde su posición actual hasta la posición deseada.

Cuando el usuario efectúa cambios en el resumen de un concepto, el sistema le ofrece la posibilidad de salvar las modificaciones realizadas en la organización del resumen. En caso afirmativo, los cambios acometidos por el usuario en un resumen concreto actualizan la estructura de composición genérica utilizada para generar sus futuros resúmenes. De este modo, se consigue una GSP del resumen adaptada a las preferencias del usuario.

Por ejemplo, si en el resumen del concepto *AHS* mostrado en la figura 4, el usuario contrae la sección *Example*, extiende la sección *Opinion* y reubica la sección *Definition* antes de la sección *Introduction*, la organización genérica de los resúmenes queda como sigue:

organización-resumen(EC_M) = { (“definición”, “obligatorio”), (“introducción”, “obligatorio”), (“ejemplo”, “opcional”), (“opinión”, “obligatorio”), (“bibliografía”, “opcional”), ...}

5.2 Necesidad de un mecanismo de generación dinámica

Aunque inicialmente los resúmenes de los conceptos son idénticos para todos los usuarios, después la organización de éstos puede ser personalizada. En este caso, cuando el usuario modifica la estructura de composición de sus resúmenes, ya no le son válidos los resúmenes pre-generados de acuerdo a la estructura de composición estándar definida por el autor.

Debido a que los resúmenes varían de un usuario a otro y a lo largo del tiempo para un mismo usuario, se hace preciso un **mecanismo de composición de resúmenes dinámico**. El mecanismo puede ser implementado de muchas formas, pero básicamente consiste en crear un nuevo hiperdocumento con tantas secciones horizontales como roles diferentes existen en la EC_M y dividir verticalmente cada sección en dos partes.

La parte izquierda de cada sección muestra el nombre del rol y un botón para contraer/expandir, reservándose la parte derecha (con mayor anchura) para presentar

información acerca del concepto. Inicialmente el botón de una sección está habilitado para "contraer" si la sección es obligatoria o para "expandir" si la sección es opcional.

Partiendo de este esqueleto general, para cada concepto, se obtiene el resumen de la siguiente manera:

- En la parte derecha de cada sección opcional se enumeran los identificadores de los ítems asociados funcionalmente con ese rol al concepto resumido.

- En las secciones obligatorias, la parte derecha se divide horizontalmente en tantas subpartes como ítems estén ligados al concepto con ese rol. Finalmente cada una de esas subpartes muestra el contenido de uno de esos ítems.

En la figura 5 se muestra parcialmente el esqueleto de un resumen, donde la organización-resumen = {(r_f^1, "obligatorio"), (r_f^2, "opcional"),...}. Observe como la sección horizontal del rol r_f^1 aparece antes que la sección destinada al rol r_f^2, y como el botón de la primera capacita para "contraer" esa sección obligatoria mientras que el de la segunda permite "expandir" la sección opcional.

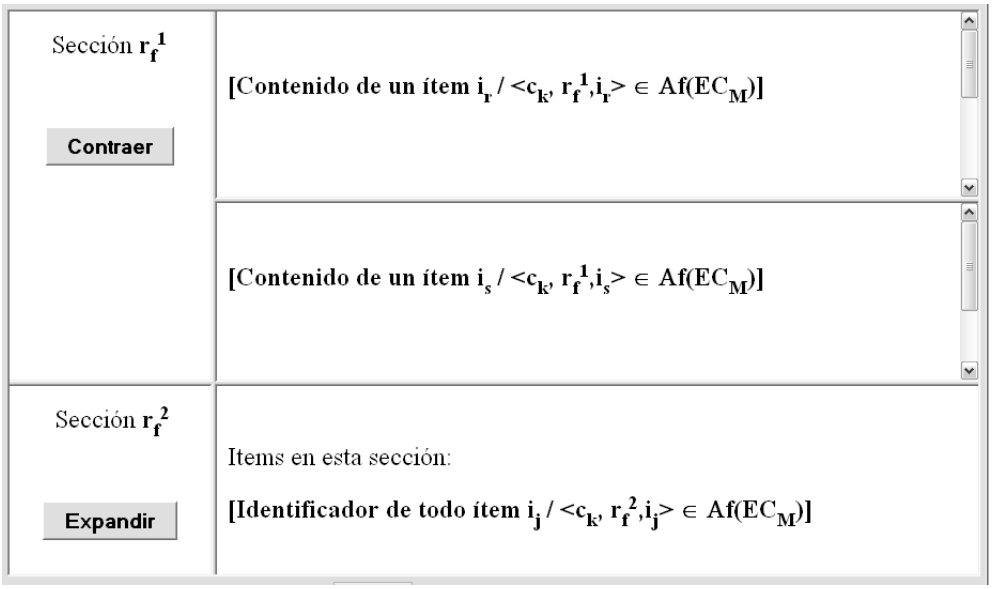


Figura 5: Esqueleto de un resumen

Advierta, también, que se han realizado las divisiones necesarias para mostrar el contenido de dos ítems en la sección del rol r_f^1. Mientras que en la sección opcional r_f^2 se requiere una sola división para expresar la lista de identificadores de los ítems asociados.

Cada vez que el sistema genera un resumen de acuerdo a una estructura de composición personalizada, lo almacena para no repetir el proceso de composición la próxima vez que ese mismo usuario solicite un resumen del mismo concepto.

Los resúmenes personalizados son almacenados para cada usuario hasta que éste modifique la estructura de composición con que fueron generados o que el autor cambie algún elemento en la EC_M que afecte al contenido de dichos resúmenes.

6. NAVEGACIÓN POR CONCEPTOS

6.1 Interfaz de navegación y actualización del modelo de usuario

Para la navegación por conceptos se proporciona al usuario una interfaz como la que se muestra en la figura 6. Como puede verse, la interfaz se divide en tres paneles o marcos:

- Un panel superior-izquierdo donde se proporciona la *estructura de navegación* que el usuario puede recorrer seleccionando conceptos.

- Un panel inferior-izquierdo que muestra *información sobre los ítems ligados funcionalmente al concepto* seleccionado. Dicha información consiste en una lista con todos los ítems asociados al concepto en la EC_M, indicando junto a cada uno el valor que posee respecto a las siguientes propiedades: identificador, nombre, rol, fecha de creación, autor, dificultad e idioma.

- Un panel derecho que visualiza el *resumen del concepto* seleccionado de acuerdo a la estructura de composición actual y a los ítems indicados en el panel anterior. En este panel el usuario puede leer el resumen del concepto que le interesa, y en caso de considerarlo oportuno, modificar su estructura al reordenar, expandir o contraer las secciones. Pudiendo, también, si así lo desea, fijar la estructura modificada como estructura de composición general para la creación de sus próximos resúmenes.

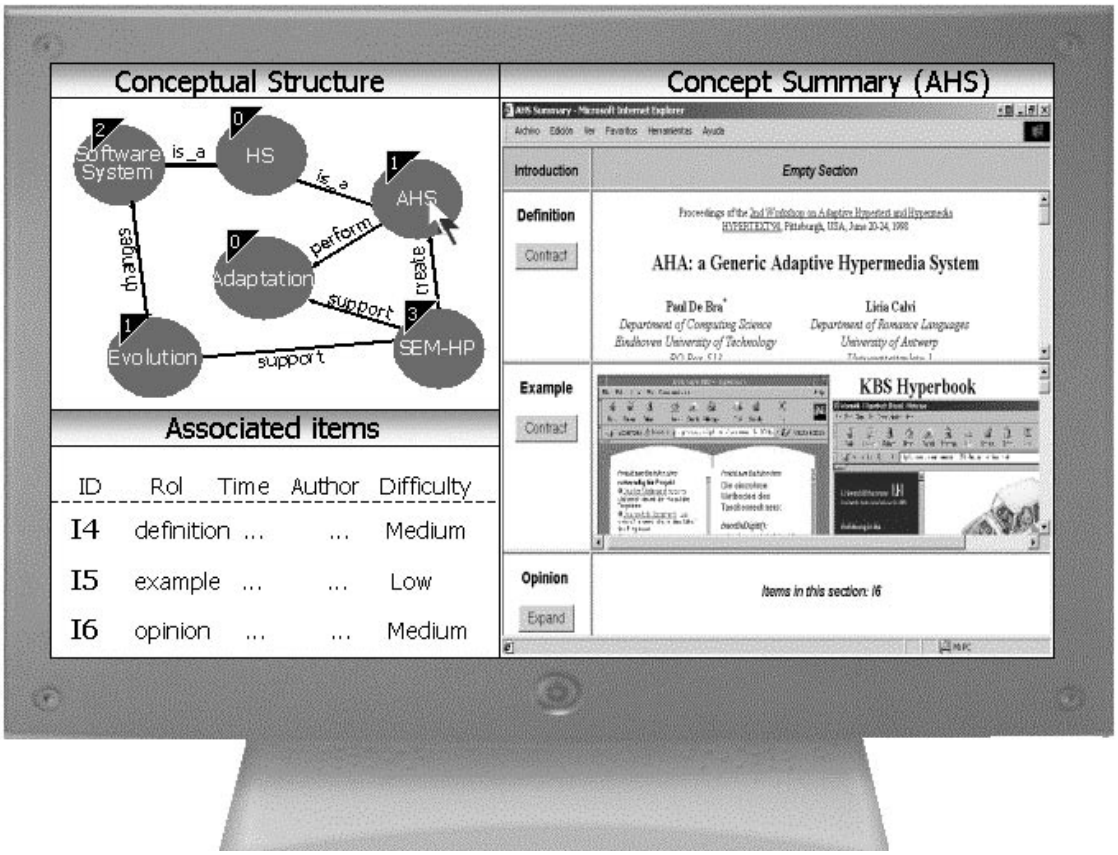


Figura 6: Interfaz para navegación por conceptos

Durante el proceso de navegación basado en conceptos, no existe ningún tipo de restricción sobre qué conceptos puede visitar el usuario. De este modo, el usuario es libre para seleccionar cualquiera de las unidades, en este caso conceptos, que se le proporcionan en la estructura de navegación.

Después de la visita del usuario a un concepto, el sistema actualiza de forma automática el modelo del usuario para capturar toda la información posible acerca de éste y su navegación. Concretamente, la actualización que se realiza cuando el usuario selecciona un concepto c_k e inspecciona su resumen tiene lugar sobre tres atributos diferentes del modelo de usuario: número de visitas, grado de conocimiento y preferencias para la estructura de los resúmenes. Las circunstancias en que cada una de estas actualizaciones tiene lugar son las siguientes:

1. Actualizar el **número de visitas** realizadas al concepto c_k:

Cada vez que el usuario accede al resumen de un concepto c_k, el número de visitas realizadas al concepto se incrementa en uno. Esto es, $visitas(c_k)$ = visitas(c_k) + 1.

2. Actualizar el **grado de conocimiento** del usuario sobre ítems y conceptos:

Cada vez que el usuario selecciona un concepto c_k, el sistema comprueba para cada ítem i_j a cuyo contenido ha tenido acceso el usuario en el resumen, si se satisfacen las restricciones de accesibilidad establecidas en sus reglas de conocimiento, y en caso afirmativo realiza las actualizaciones pertinentes sobre el conocimiento del usuario.

El conjunto de ítems cuyo contenido se muestra en el resumen de c_k, está formado por los ítems i_j incluidos en elementos-resumen(c_k) que están asociados al concepto c_k mediante un rol r_f^j cuya sección ha estado en algún momento expandida en el resumen. Se incluyen pues, los ítems asociados a c_k con un rol inicialmente "obligatorio" o con un rol "opcional" expandido por el usuario durante la lectura del resumen. En definitiva, todos los ítems mostrados desde que el usuario obtiene el resumen de c_k hasta que selecciona otro concepto[1].

Para cada ítem i_j visualizado en el resumen, el sistema comprueba si el usuario estaba preparado para asimilarlo, validando sobre el estado de conocimiento de éste las reglas de conocimiento Rk(i_j) definidas en la estructura de aprendizaje actual. En caso positivo, el sistema ejecuta la regla de actualización asociada, Ru(i_j), modificando el grado de conocimiento de i_j, $K(i_j)$, y quizá de otros ítems relacionados.

Una vez que se ha actualizado el grado de conocimiento de todos los ítems que corresponda, el sistema actualiza el grado de conocimiento del concepto c_k, $K(c_k)$, aplicando su regla de peso, Rw(c_k).

3. Actualizar la **estructura de composición** de los resúmenes tras las modificaciones realizadas en el resumen actual:

[1] Otra opción posible es considerar únicamente los ítems mostrados en el resumen final de c_k, esto es, justo antes de que el usuario seleccione el siguiente concepto.

Esta actualización es ejecutada cuando el usuario ha modificado la organización del resumen actual y desea que esos cambios sean aplicados de forma general en la estructura de composiciones de sus resúmenes.

6.2 Técnicas de adaptación utilizadas

Tal y como se puede ver en la figura 6, la única adaptación que se lleva a cabo en la estructura de navegación consiste en anotar sobre ésta el número de visitas realizadas a cada concepto. Concretamente se coloca junto a cada concepto c_k, enmarcado dentro de un triángulo superior, el número de veces que éste ha sido visitado por el usuario. Esta información se obtiene directamente del atributo **visitas(c_k)** almacenado en el modelo del usuario, y evita, salvo deseo expreso, que el usuario repita la visita de un concepto.

La adaptación realizada durante este modo de navegación se centra en adaptar la presentación de la información proporcionada y no la navegación de ésta. De forma más precisa, podemos decir que se trata de personalizar la presentación de los resúmenes generados.

Para determinar la apariencia del resumen de un concepto, el sistema aplica técnicas adaptativas que permiten personalizar la información contenida en el resumen a las preferencias del usuario. Concretamente, teniendo en cuenta la organización del resumen elegida por el usuario y el conjunto de ítems asociados al concepto seleccionado, el sistema genera el resumen aplicando tres técnicas basadas en *frames* [Brusilovsky, 96]:

- **Texto condicional**: La información sobre un concepto está dividida en distintas piezas de información denominadas ítems. Cada ítem tiene asociado un rol o papel funcional. Cada rol es designado como “obligatorio” u “optativo” en la estructura de composición de los resúmenes almacenada en el modelo de usuario. Cuando se visualiza el resumen de un concepto sólo se muestra el contenido de los ítems asociados con un rol “obligatorio”.

- **Texto desplegable** (*strechtext*): El carácter “obligatorio” u “opcional” de un rol, determina si su correspondiente sección en el resumen aparece inicialmente extendida o contraída. El usuario puede modificar esta propiedad, al expandir o contraer las secciones del resumen. De esta manera, aunque el resumen contenga mucha información, el propio usuario puede mostrar u ocultar sus secciones, de manera que finalmente visualice únicamente las que le interesan.

- **Ordenación**: No sólo se personaliza el carácter extendido/contraído de las secciones. El usuario puede decidir, también, en qué orden se colocan las secciones en el resumen, para hacer su lectura lo más agradable y provechosa posible.

7. NAVEGACIÓN POR CONCEPTOS SOBRE UNA EC_P

Si bien, el modo de navegación por conceptos, es especialmente interesante cuando la estructura de navegación se obtiene a partir de la estructura conceptual de memorización completa, también puede aplicarse sobre cualquiera de las presentaciones creadas a partir de ella.

Aunque las estructuras conceptuales de presentación han sido concebidas para tener un número reducido de elementos, el autor puede verse tentado de crear presentaciones con un tamaño considerable, candidatas a ocasionar problemas de desorientación. Por lo tanto, si una vez elegida la estructura de aprendizaje que mejor se ajusta al usuario, la dimensión de la presentación EC_P^i incluida en ésta es excesiva, la capacidad de navegarla por conceptos puede ser muy provechosa para el usuario.

En este caso, la estructura de navegación no se corresponde con el dominio conceptual completo sino con una parcela delimitada de éste. Es decir, la red semántica ofrecida para navegar muestra únicamente los conceptos y relaciones conceptuales existentes en la presentación elegida, esto es $C(EC_P^i)$ y $Ac(EC_P^i)$ respectivamente.

La interfaz de navegación y la actualización del modelo de usuario coinciden con lo especificado a lo largo del presente capítulo para la EC_M. La generación de los resúmenes también se realiza de acuerdo a lo descrito. Salvo que, en este caso, aunque se utiliza la misma organización del resumen, esto es, se mantiene organización-resumen(EC_M). Para instanciar el resumen de un concepto concreto, c_k, el conjunto de elementos implicados, elementos-resumen(c_k), es diferente.

Esto se debe a que obviamente, para componer el resumen de un concepto c_k no se consideran todos los ítems asociados a éste en la EC_M, sino sólo los que se capturan en la presentación actual. Esto es, el conjunto de ítems que integran el resumen se define ahora como sigue:

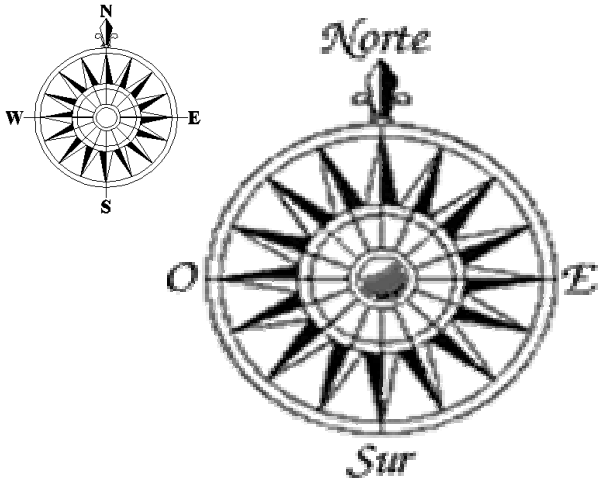
$$i_j \in \text{elementos-resumen}(c_k) \text{ si } \exists\, a_f \in Af(EC_P^i) \text{ tal que } a_f = <c_k, r_f, i_j> \quad (1')$$

En definitiva, el modo de navegación por conceptos sobre una EC_P^i es idéntico al acontecido sobre la EC_M, salvo en lo que respecta a los conceptos y relaciones conceptuales incluidas en la estructura de navegación, y a los ítems y asociaciones funcionales utilizadas en la instanciación de los resúmenes. En pocas palabras, la navegación por conceptos sobre una EC_P^i se realiza igual que sobre la EC_M pero acorde con el subdominio conceptual y de información capturados.

Por otra parte, la posibilidad de realizar navegación por conceptos sobre las estructuras conceptuales de presentación, viene a reforzar la necesidad de un mecanismo dinámico para generar los resúmenes. Sería muy poco eficiente generar a priori para cada usuario (de acuerdo a su organización-resumen personalizada) el resumen de un concepto para cada una de las presentaciones que lo incluye. Generando bajo demanda cada uno de estos resúmenes, el sistema elimina el esfuerzo de preparar de antemano todos los resúmenes posibles, ya que muchos de éstos seguramente nunca serán solicitados.

CAPÍTULO 17

Navegación por Relación Conceptual



Resumen

Una descripción detallada del modo de navegación por relación conceptual es realizada en el presente capítulo. Se establecen sus características básicas y se especifican las restricciones tenidas en cuenta para dirigir la navegación del usuario en un orden coherente con las relaciones semánticas definidas en el dominio conceptual. Se expone un procedimiento para identificar cuándo un ítem puede ser visitado por el usuario, y un conjunto de técnicas de adaptación para personalizar, en consecuencia, la estructura de navegación proporcionada. También se especifica el proceso de actualización que tiene lugar sobre el modelo del usuario a medida que éste navega en el modo descrito.

Tabla de contenidos

Navegación por Relación Conceptual

1. INTRODUCCIÓN

Hasta ahora, los modos de navegación explicados, tradicional (capítulo 15) y por conceptos (capítulo 16), permiten al usuario acceder a cualquiera de las unidades de navegación disponibles en la estructura proporcionada. Se trata de un proceso de navegación libre, donde cada vez que el usuario selecciona una unidad de navegación obtiene el contenido de la misma; ya sea la información de un ítem en la navegación tradicional o el resumen de un concepto en la navegación por conceptos.

Por el contrario, la navegación por relación conceptual es un **modo de navegación restringido**, lo cual implica que no se satisface cualquier petición del usuario. Es decir, únicamente si le está permitido el acceso a una unidad de navegación, su selección va a permitirle inspeccionar el contenido de ésta. De manera que, cuando el usuario selecciona una unidad de navegación no permitida, dicha selección no produce efecto alguno.

En este caso, al igual que en la navegación tradicional, la estructura de navegación proporcionada coincide con la red semántica que representa la **estructura conceptual de presentación** asociada a la estructura de aprendizaje elegida para el usuario.

La **unidad de navegación** propia de este modo **es el ítem**, de forma que el usuario recorre la información visitando de forma individual los distintos ítems que han sido incluidos en esa presentación.

> **Def 17.1 [Navegación por relación conceptual]** La navegación por relación conceptual es un modo de navegación restringido, que define *una forma de recorrer la estructura de navegación coherente con las relaciones semánticas existentes entre sus conceptos*. La unidad de navegación es el ítem y la red semántica proporcionada para navegar coincide con la EC_P^k incluida en la EC_A^j elegida para el usuario.

En cada momento, el siguiente paso de navegación de un usuario está limitado a la selección de algún ítem ligado a un **concepto disponible** desde el **concepto actual**. El concepto actual, c_a, es aquél al que se asocia funcionalmente el último ítem inspeccionado por el usuario. Un concepto c_b está disponible si existe una relación conceptual $r_c{:}c_a \rightarrow c_b$ que puede ser navegada desde el concepto actual hasta él.

2. RESTRICCIONES DE NAVEGACIÓN

La navegación por relación conceptual, como su propio nombre indica, tiene en cuenta las relaciones entre conceptos que aparecen en la estructura de navegación. Más específicamente, depende de dos elementos que el autor define en la fase de navegación sobre las relaciones conceptuales:

1) las **restricciones de navegabilidad** (RTnb), y

2) las **reglas de orden** (Ro).

Ambos elementos forman parte del Sistema de Navegación [García, 01c], y son utilizados por el Sistema de Aprendizaje para restringir y adaptar un proceso de navegación que tiene lugar a través de las relaciones conceptuales incluidas en la estructura de navegación.

En cada paso de navegación, el sistema determina las *relaciones que puede recorrer el usuario* en función del concepto actual y las reglas de navegabilidad definidas sobre las relaciones que involucran a dicho concepto en la estructura actual.

Una vez establecidas las relaciones que el usuario puede seguir, es inmediato obtener el subconjunto de conceptos a los que el usuario puede llegar en un sólo paso. Las reglas de orden permiten establecer cuáles de los ítems asociados a estos conceptos son susceptibles de ser inspeccionados por el usuario, y por lo tanto, su selección produce el efecto deseado.

Obviamente, el conjunto de reglas utilizadas para controlar la navegación de un usuario son las que se encuentran dentro de la estructura de aprendizaje elegida para éste. Esto es, siendo EC_A^j = (EC_M, Rw, MU, EC_P^k, Ro^i, $RTnb^i$, Ru^j, Rk^j) la estructura de aprendizaje que mejor se ajusta a la experiencia y subdominio de interés del usuario, el conjunto de reglas de orden y navegabilidad que condicionan la navegación del usuario en este modo son $\mathbf{Ro^i}$ y $\mathbf{RTnb^i}$.

2.1 Restricciones de navegabilidad

Las restricciones de navegabilidad, $RTnb^i$, se definen sobre una estructura conceptual de presentación EC_P^k y determinan en qué dirección pueden navegarse las relaciones conceptuales que en ella aparecen. Por defecto, una relación conceptual puede navegarse únicamente en el sentido de la flecha, esto es, desde el concepto origen al concepto destino. Sin embargo, el autor puede ampliar la **navegabilidad** de una relación conceptual en los dos sentidos, esto es desde el origen al destino y desde el destino al origen.

En la figura 1 se muestra la trasformación gráfica de una relación conceptual cuya navegabilidad ha sido ampliada por el autor para que puede ser navegada en los dos sentidos.

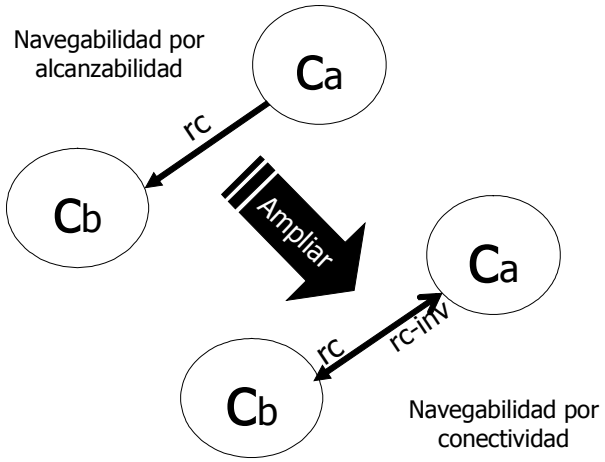


Figura 1: Navegabilidad de una relación conceptual

Como se observa en la figura 1, en el primer caso la navegabilidad viene determinada por la **alcanzabilidad**, esto es, desde el concepto c_a se puede llegar al concepto c_b sólo si existe una relación conceptual r_c: $c_a \rightarrow c_b$. Mientras que, en el segundo caso, la navegabilidad está determinada por la **conectividad**, es decir, si existe una relación r_c: $c_a \rightarrow c_b$ se puede alcanzar c_a desde c_b y c_b desde c_a. Ya que para el sistema es como si existiese una relación conceptual inversa r_c-inv: $c_b \rightarrow c_a$.

Una vez establecida la navegabilidad de las relaciones conceptuales, éstas pueden verse como enlaces de navegación, que deben seguirse para pasar de un concepto a otro. El carácter semántico de estos enlaces juega un papel importante en el proceso de aprendizaje del usuario, ya que evita transiciones conceptuales bruscas en el recorrido que éste realiza a través de la información. Esto es, si el usuario lee un ítem i_j e inmediatamente después lee otro ítem i_g, se garantiza que existe algún tipo de relación entre los conceptos sobre los que versan ambos ítems.

2.2 Reglas de orden

Las reglas de navegabilidad permiten establecer qué conceptos de la estructura de navegación están disponibles dependiendo de cuál sea el concepto actual. Sin embargo, en este modo, la unidad de navegación es el ítem y no el concepto. Por lo que es necesario, algo más que discrimine para cada concepto disponible qué ítems pueden ser visitados y cuáles no. Como ya se ha adelantado, ese algo más son las reglas de orden.

Para cada estructura conceptual de presentación EC_P^k se define un conjunto de reglas de orden $\mathbf{Ro^i}$ que establecen las circunstancias en las que desde un ítem se puede ir hacia otro, en función de:

a) la existencia y navegabilidad de las relaciones conceptuales entre los conceptos a los que dichos ítems se asocian, y

b) los ítems visitados anteriormente.

El punto **a)** es **fijo** y determina que la transición de un ítem $c_n.i_j$ a otro $c_m.i_r$ sólo es posible si existe un enlace de navegación que puede ser navegado desde el concepto c_n hasta el concepto c_m. Es decir, existe en la presentación una relación conceptual que va desde c_n hasta c_m o una que va desde c_m hasta c_n cuya navegabilidad ha sido ampliada en el otro sentido. Véase la ecuación 1.

$$r_c{:}c_n \rightarrow c_m \in Ac(EC_P^k) \vee [\, rc{:}c_m \rightarrow c_n \in Ac(EC_P^k) \wedge rc\text{-}inv{:}c_n \rightarrow c_m \in Rnb^i(EC_P^k) \,] \qquad (1)$$

El punto **b)** es **opcional** y permite al autor reflejar la necesidad de haber visitado anteriormente uno o varios ítems, para poder inspeccionar el contenido de un ítem $c_m.i_r$.

Tanto en a) como en b) se establecen relaciones de orden en la visita de los ítems, pero de forma ligeramente diferente. De lo establecido en el punto **a)** se deduce que para acceder a $c_m.i_r$ el usuario debe haber visitado inmediatamente antes un ítem asociado al concepto c_n. Se trata, pues, de una **visita previa** a un ítem $c_n.i_j$ enlazado con el ítem $c_m.i_r$.

Por el contrario en **b)** el autor puede establecer que para acceder a $c_m.i_r$ es necesario haber visitado el ítem $c_s.i_t$, aunque no exista relación directa entre los conceptos c_s y c_m. Tratándose en este caso, de una visita necesariamente **anterior** en dos o más pasos.

En resumen, podemos decir que en a) se especifica el concepto actual a través de un conjunto de ítems de los cuales uno debe ser el último ítem visitado, y en b) se dice que ítems deben haberse inspeccionado alguna vez en el pasado.

2.2.1 Estructura de una regla de orden

Asociado a cada ítem de una presentación EC_P^k existe un conjunto de reglas de orden que determinan el acceso a dicho ítem en función del recorrido realizado por el usuario. El conjunto de reglas definido para el ítem $c_m.i_r$ se denomina **$Ro(c_m.i_r)$** y puede contener tantas reglas de orden como enlaces de navegación tienen su destino en el concepto c_m, esto es $Ro(c_m.i_r) = \{Ro^1(c_m.i_r), Ro^2(c_m.i_r), ...\}$. Todas las reglas incluidas en $Ro(c_m.i_r)$ se definen sobre el mismo ítem, $c_m.i_r$, pero cada una implica a una relación conceptual r_c distinta que, por supuesto, debe ser navegable hacia c_m.

Las restricciones de orden incluidas en una regla, se expresan usando **lógica temporal proposicional**. Concretamente, se utilizan tres predicados:

- **seguir(r_c)** para indicar la obligación de atravesar la relación conceptual r_c.

- **previo($c_n.i_j$)** para exigir una visita inmediatamente anterior al ítem i_j que se asocia al concepto c_n.

- **antes(i_j)** para requerir sobre el ítem i_j una visita anterior en al menos dos pasos.

Observe que en el predicado *antes* no es necesario especificar el concepto al que se asocia el ítem, mientras que en el predicado *previo* sí es requerido. Esto se debe a que, acceder a un ítem $c_m.i_r$ realizando una visita previa al ítem i_j sólo es posible si éste se asocia a un concepto c_n desde el que parte una relación navegable hasta c_m. Por lo tanto, es preciso conocer el concepto al que se asocia i_j, distinguiendo entre uno y otro, en el caso de que el ítem se asocie a varios conceptos.

De acuerdo a lo establecido en la sección 2.2, una regla de orden se divide en dos partes: una obligatoria que es generada automáticamente por el sistema (a) y una opcional escrita por el autor (b). Estas dos partes se unen mediante el operador lógico *and* tal y como se muestra en la ecuación 2.

$$Ro(c_m.i_r): \text{fórmula-sistema and [fórmula-autor]} \qquad (2)$$

La parte obligatoria se denomina **fórmula-sistema** y especifica qué relación conceptual se sigue. Dicha relación, r_c: $c_n \rightarrow c_m$, debe poder recorrerse desde algún concepto c_n hasta el concepto c_m al que está ligado el ítem cuya visita se restringe en la regla. Al exigir el seguimiento de la relación r_c: $c_n \rightarrow c_m$ se está estableciendo como concepto actual c_n, y requiriendo una visita previa a alguno de los *s* ítems, $c_n.i_1$, $c_n.i_2$,..., $c_n.i_s$, ligados a éste en la presentación actual. Véase la ecuación 3.

$$\text{fórmula-sistema: seguir}(r_c{:}c_n \rightarrow c_m) \text{ and [previo}(c_n.i_1) \text{ or } \ldots \text{ or previo}(c_n.i_s)]$$

$$\text{, donde } \forall_{j:1..s} <c_n, r_f, i_j> \in Af(EC_P^k) \qquad (3)$$

La parte opcional se denomina **fórmula-autor** y permite a éste añadir restricciones de tipo *antes* sobre cualquier ítem que desee. Véase la ecuación 4.

$$\text{fórmula-autor: antes}(i_u) \text{ and } ... \text{ and antes}(i_v), \text{ donde } i_u, ..., i_v \in I(EC_P^k) \quad (4)$$

2.2.2 Generación de las reglas de orden

Por defecto para cada ítem $c_m.i_r$ de la EC_P^k existe una regla de orden por cada relación conceptual, r_c, que pueda ser navegada hasta el concepto c_m al que se asocia funcionalmente el ítem considerado. En todas las reglas por defecto, la fórmula-autor es vacía y la fórmula-sistema se construye según lo especificado en la ecuación 3.

En el ejemplo de la figura 2, se muestra el conjunto de reglas de orden generadas por defecto (figura 2.2) para una estructura conceptual de presentación (figura 2.1).

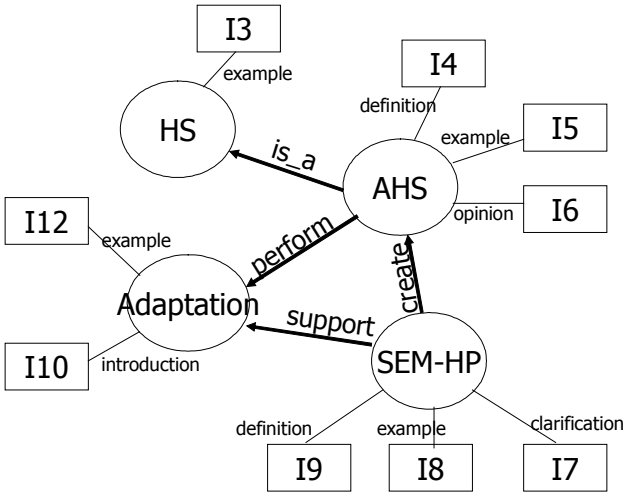


Figura 2.1: Una EC_P

Ro(I3)	Ro^1(I3): seguir(is_a) and [previo(I4) or previo(I5) or previo(I6)]
Ro(I4)	Ro^1(I4): seguir(create) and [previo(I7) or previo(I8) or previo(I9)]
Ro(I5)	Ro^1(I5): seguir(create) and [previo(I7) or previo(I8) or previo(I9)]
Ro(I6)	Ro^1(I6): seguir(create) and [previo(I7) or previo(I8) or previo(I9)]
Ro(I7)	∅
Ro(I8)	∅
Ro(I9)	∅
Ro(I10)	Ro^1(I10):seguir(perform) and [previo(I4) or previo(I5) or previo(I6)] Ro^2(I10):seguir(support) and [previo(I7) or previo(I8) or previo(I9)]
Ro(I12)	Ro^1(I12):seguir(perform) and [previo(I4) or previo(I5) or previo(I6)] Ro^2(I12):seguir(support) and [previo(I7) or previo(I8) or previo(I9)]

Figura 2.2: Reglas de orden por defecto para la EC_P

Por simplicidad utilizamos para referirnos a un ítem dentro de una regla de orden únicamente su identificador (I3, I4,...). Sin embargo, esto sería ambiguo en el caso de

que un mismo ítem estuviese asociado funcionalmente a dos conceptos, siendo lo correcto HS.I3, AHS.I4, AHS.I5, etc.

Observe como el número de reglas definidas para un ítem coincide con el número de relaciones que llegan al concepto al que éste se asocia: cero en el caso de I7, I8 e I9, dos para I10 e I12 y una para el resto de ítems.

El conjunto de reglas de orden asociadas a un ítem contempla los distintos caminos que el usuario puede seguir para llegar hasta él y depende del concepto al que está asociado. El autor puede restringir, aún más, cada uno de esos caminos imponiendo nuevas restricciones de orden en su fórmula-autor, que por defecto está vacía.

Estas restricciones siempre son de tipo *antes*, es decir, se trata de visitas que el usuario debe haber realizado hace dos o más pasos de navegación. Y sirven al autor para exigir la visita de uno o varios ítems que el considera, de alguna forma, necesarios para poder acceder al ítem actual.

Como puede verse en la figura 2.2, el conjunto inicial de reglas de orden es idéntico para todos los ítems asociados a un mismo concepto. De modo que será el autor, si así lo desea, el que establezca nuevas restricciones que hagan diferente el camino a dos ítems del mismo concepto.

Por ejemplo, el autor puede modificar la regla asociada al ítem I10 (figura 2.2), para que exija una visita anterior al ítem I3 cuando se pretende llegar a éste siguiendo la relación conceptual *support*.

Tras esta modificación, la regla queda como se muestra en la tabla 1.

Tabla 1: Regla de orden modificada

Ro^2(I10): seguir(support) and [previo(I7) or previo(I8) or previo(I9)] **and antes(I3)**

3. DETERMINAR ÍTEMS ACCESIBLES

En la navegación por relación conceptual las reglas de orden son evaluadas para determinar qué ítems puede visitar el usuario a continuación. Para ello, el sistema necesita conocer el concepto actual, es decir al que se asocia el último ítem visitado y las visitas realizadas anteriormente por el usuario. Esta última información se encuentra almacenada y actualizada en el modelo del usuario, concretamente en el atributo *visitas(i_j)* asociado a cada ítem del sistema (capítulo 12).

Conociendo únicamente el concepto actual, el sistema puede **evaluar la fórmula-sistema** de una regla de orden. Puesto que todos los ítems asociados a un mismo concepto coinciden en la fórmula-sistema de sus reglas de orden, se ahorra esfuerzo y tiempo si se evalúa ésta a nivel de concepto, en lugar de para cada ítem. De este modo, inicialmente el sistema identifica los *conceptos disponibles*, como aquellos para los que el concepto actual satisface la fórmula-sistema.

Suponiendo que el último ítem visitado cuelga del concepto c_n, un concepto c_m está disponible si:

a) c_m no es destino de ninguna relación conceptual. En este caso la fórmula-sistema es vacía y por defecto *true*.

b) c_m es el concepto actual ($c_m = c_n$). En este caso, al no ser necesario ningún cambio de concepto no se reevalúa la fórmula-sistema.

c) Existe un enlace de navegación desde el concepto actual c_n hasta c_m. Esto es, $r_c{:}c_n \rightarrow c_m$ o $r_{c\text{-}inv}{:}c_n \rightarrow c_m$.

Def 17.2 [*Starting-point*] Dada una presentación EC_P^k y un conjunto de reglas de navegabilidad $RTnb^i$, denominamos punto de partida o *starting-point* a un concepto que no es destino de ningún enlace de navegación y que por lo tanto siempre está disponible durante la navegación por relación conceptual, independientemente de cuál sea el último concepto visitado.

Todos los ítems asociados a un concepto *starting-point* son accesibles en cualquier momento durante la navegación por relación conceptual, ya que su conjunto de reglas de orden es vacío. Siempre debe haber al menos uno, ya que, tal y como su nombre indica, cuando un usuario inicia su navegación en una presentación, únicamente puede visitar los ítems asociados a un *starting-point*.

Volviendo al ejemplo de la figura 2, vemos que los ítems I7, I8 e I9 no tienen ninguna restricción de orden asociada. Esto se debe a que el concepto *SEM-HP*, al que se asocian funcionalmente, es un *starting-point*. Además, al ser el único que existe en la presentación, constituye el inevitable punto de partida para la navegación de ésta.

Una vez obtenido el conjunto de conceptos que están disponibles en el siguiente paso de navegación, el sistema prohíbe el acceso a cualquier ítem asociado a un concepto no disponible. Y, a continuación, procede a **evaluar la fórmula-autor** de los ítems asociados a conceptos disponibles, determinando en función de las visitas anteriores del usuario a cuáles de ellos puede acceder y a cuáles no.

Tal y como se expuso anteriormente, determinar si el usuario satisface una restricción de autor *antes(i_j)* equivale a comprobar si el usuario ha visitado con anterioridad el ítem i_j. Esta labor es automáticamente realizada, al comparar con 0 el valor del atributo *visitas(i_j)* almacenado en el modelo del usuario. De modo que, en caso de igualdad, el predicado *antes(i_j)* es evaluado como *false* y en otro caso como *true*.

Las restricciones de autor se evalúan para cada ítem asociado a un concepto disponible, de forma ligeramente distinta, según el motivo por el que el concepto ha sido identificado como tal:

a) Un ítem asociado a un concepto *starting-point* siempre puede visitarse, ya que carece de restricciones de autor asociadas.

b) Para determinar si un ítem asociado al concepto actual puede visitarse, no hay que considerar todas sus reglas de orden, sino sólo aquella que sigue la relación conceptual recorrida en el paso previo de navegación para llegar al concepto actual.

Supongamos que, sobre la EC_P de la figura 2.1, el usuario visita inicialmente el ítem I9 asociado al concepto *SEM-HP*. Después atraviesa la relación "support" y visita el ítem I12 asociado al concepto *Adaptation*.

La pregunta es, ¿puede visitar a continuación el ítem I10 asociado al concepto actual *Adaptation*? Para responder a esta cuestión, el sistema debe evaluar únicamente la fórmula-autor de la regla de orden asociada a I10 para la relación "support". Puesto que en ésta, Ro^2(I10) (tabla 1), se exige una visita anterior a I3 que no ha tenido lugar, la respuesta sería negativa.

c) Un ítem asociado a un concepto c_m alcanzable desde el concepto actual c_n, puede visitarse si se satisface la fórmula-autor de su regla de orden asociada a la relación conceptual que enlaza c_n con c_m. Si existen varias relaciones navegables desde c_n hasta c_m basta con que se satisfaga la fórmula-autor de la regla asociada a una de ellas.

Por ejemplo, si el concepto actual en la EC_P de la figura 2.1 es *SEM-HP*, el usuario puede visitar los ítems I4, I5 e I6 siguiendo la relación "create", y el ítem I12 siendo la relación "support". De nuevo el ítem I10, aunque asociado a un concepto disponible, no puede ser visitado porque no se satisface su fórmula-autor.

4. ADAPTACIÓN DE LA ESTRUCTURA DE NAVEGACIÓN

La interfaz que el usuario utiliza para navegar en este modo coincide con la proporcionada durante la navegación tradicional, esto es, dos paneles verticales: el izquierdo para visualizar la estructura de navegación y el derecho para mostrar el contenido del último ítem seleccionado.

En este caso, al igual que en la navegación tradicional, la estructura de navegación coincide con la red semántica que representa la presentación EC_P^k incluida en la estructura de aprendizaje elegida para el usuario, EC_A^j. La diferencia radica en que, en la navegación por relación conceptual, no todos los ítems de la estructura de navegación están siempre accesibles. Es decir al seleccionar un determinado ítem puede que su contenido no se muestre en el panel derecho, cosa que siempre ocurre en la navegación tradicional.

El conjunto de ítems accesibles en un momento dado depende del último ítem visitado, pero también de la navegación realizada hasta ese instante por el usuario, por lo tanto, varía de un usuario a otro. Para personalizar la funcionalidad de la estructura de navegación para cada usuario y paso de navegación, es necesario realizar un proceso de adaptación que cumpla tres objetivos:

i) identificar los ítems que el usuario puede visitar en un momento dado,

ii) impedir el acceso a los ítems no accesibles, e

iii) informar al usuario de su propio proceso de navegación.

Con la finalidad de satisfacer el **objetivo i)**, el sistema realiza un proceso de anotación sobre la estructura de navegación que permite diferenciar, en cada momento, los conceptos disponibles de los no disponibles, y los ítems accesibles de los no accesibles.

Para ello, se *resaltan* positivamente los conceptos disponibles, esto es, el concepto actual, los *starting-points* y los conceptos alcanzables desde el concepto actual. En principio todos los ítems ligados a un concepto marcado como disponible pueden ser visitados, salvo aquellos que expresamente hayan sido marcados negativamente. Dado un concepto disponible únicamente se marcan de forma negativa los ítems que no satisfacen la fórmula-autor según lo especificado en la sección anterior.

Para satisfacer el **objetivo ii)** los ítems que el usuario no está autorizado a visitar son *deshabilitados*, de forma que aunque éste los seleccione no obtiene respuesta alguna, es decir, al hacer *clic* sobre el ítem no se visualiza la "página" de destino en el panel derecho del navegador. Dentro de los ítems inaccesibles se incluyen los que se asocian a un concepto sin resaltar, o sea a un concepto no disponible, y los que a pesar de estar asociados a un concepto disponible han sido marcados negativamente.

A continuación se muestra la anotación realizada sobre la estructura de navegación de la figura 2, de acuerdo a las reglas de orden generadas por defecto, con la única modificación que se propone en la tabla 1.

En el ejemplo, se ha utilizado la figura de un peatón en movimiento sobre un fondo verde para resaltar positivamente un concepto disponible. De modo contrario, la figura de un peatón detenido sobre un fondo rojo resalta negativamente los ítems inaccesibles que se asocian a un concepto disponible.

Inicialmente, tal y como se muestra en la figura 3.a, el único concepto disponible es *SEM-HP*, el cual aparece resaltado positivamente. Al tratarse de un *starting-point* todos sus ítems son accesibles. Por claridad los ítems accesibles asociados a un concepto disponible se sombrean en color verde, en este caso I7, I8 e I9.

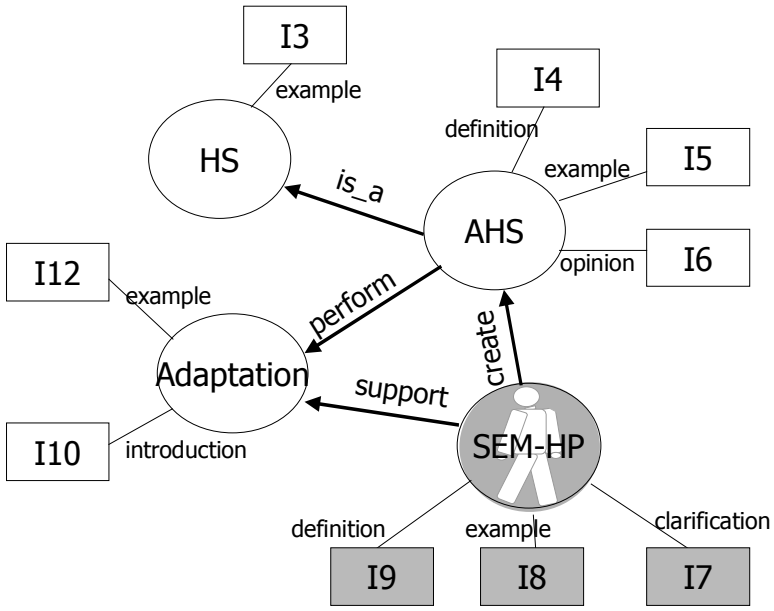


Figura 3.a: Anotación inicial

En este momento, el usuario puede visitar únicamente I7, I8 o I9. Si suponemos, por ejemplo, que selecciona el ítem I8, la anotación correspondiente es la que se muestra en la figura 3.b.

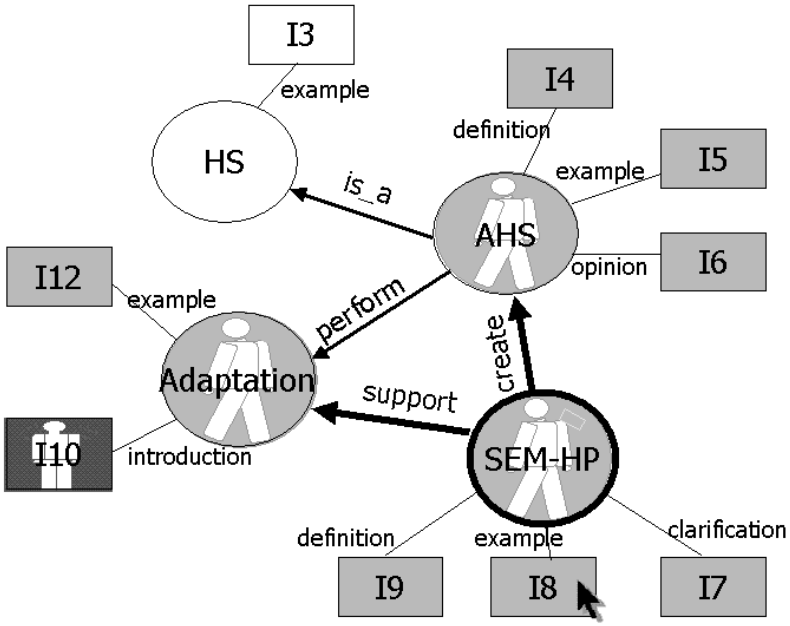


Figura3.b: Anotación tras visitar I8

Observe que el concepto actual, *SEM-HP*, y las relaciones conceptuales que parten de él, "support" y "create", se resaltan con un grosor mayor. Esto permite al usuario conocer los enlaces de navegación que puede seguir a continuación.

El concepto actual sigue estando disponible. Pero además, el destino de cada relación resaltada, *Adaptation* por "support" y *AHS* por "create", es también un concepto disponible y se resalta como tal.

Advierta que el ítem I10 asociado al concepto *Adaptation* aparece marcado como inaccesible, ya que el usuario no satisface la fórmula-autor (visitar antes I3) de la regla de orden que permite visitarlo siguiendo la relación "support".

De acuerdo a la anotación realizada, el usuario sabe que, en este paso, puede visitar I4, I5, I6, I7, I8, I9 o I12. No obstante, si aún así, selecciona I10 o I3 no se produce efecto alguno.

Puesto que el proceso de navegación depende de las visitas realizadas anteriormente por el usuario y el orden en que éstas se producen, es conveniente anotar sobre la estructura de navegación el número de veces que el usuario ha visitado cada ítem (**objetivo iii**). El sistema consulta esta información en el modelo del usuario, concretamente en el atributo *visitas(i_j)*.

En este modo de navegación, la anotación del número de visitas consigue un doble beneficio. Por un lado, el usuario sabe inmediatamente si ha visitado anteriormente un ítem sin necesidad de abrirlo para inspeccionar su contenido. Esto le evitará perder tiempo al visitar repetidas veces un ítem por olvido o despiste. Por otro lado, la propia estructura de navegación visualiza la información necesaria para evaluar las reglas de

orden, lo que hace más consciente al usuario de las restricciones de navegación existentes y lo que debe hacer para superarlas (especialmente en sistemas donde las reglas de navegación son visibles para el usuario).

Observe en el ejemplo de la figura 4, la anotación de un número junto a cada ítem (triángulo en la esquina superior derecha), contabilizando las veces que el usuario ha seleccionado dicho ítem.

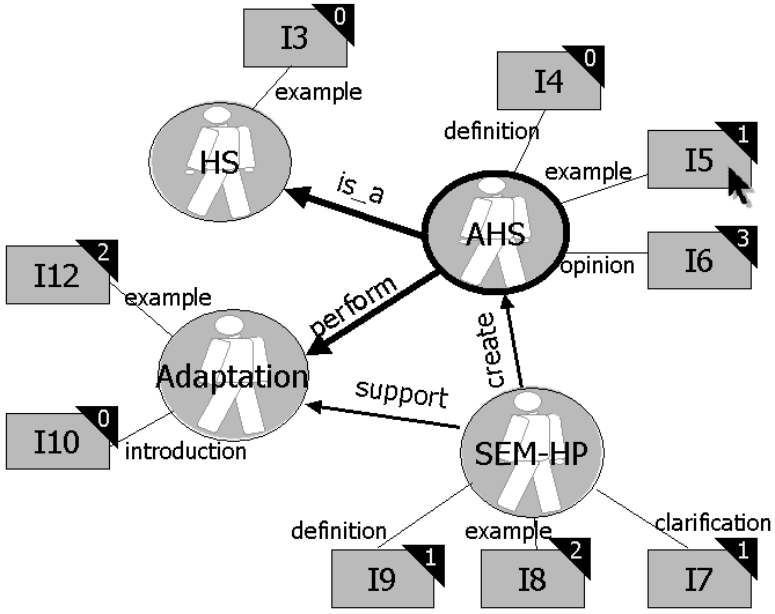


Figura 4: Anotación de visitas sobre la EC_p

Advierta, además, que siendo el concepto actual *AHS*, todos los conceptos están disponibles y todos los ítems son accesibles. Ahora, el ítem I10 sí puede visitarse, aunque el número de visitas sobre I3 sigue siendo cero. Esto se debe a que la regla de orden asociada a I10 que ahora se evalúa, es la que implica el seguimiento de la relación "perform" o sea $Ro^1(I3)$ (figura 2.2), y en ésta no existe ninguna restricción de autor.

5. ACTUALIZACIÓN DEL MODELO DE USUARIO

A medida que el usuario navega, el conjunto de ítems que le está permitido visitar varía en función del concepto actual, los enlaces de navegación que parten de él y las restricciones de orden impuestas por el autor sobre los ítems asociados a los conceptos destino.

Por tanto, en este modo de navegación, la definición de accesibilidad utilizada para permitir o no la visita a un ítem es proporcionada por las reglas de orden, en lugar de por las reglas de conocimiento. Es decir, un ítem $c_k.i_j$ es accesible si la navegación realizada por el usuario satisface una de sus reglas de orden, $Ro(c_k.i_j)$, independientemente de si el estado de conocimiento del usuario cumple los requisitos de accesibilidad impuestos en alguna de sus reglas de conocimiento, $Rk(i_j)$.

De esta manera, la accesibilidad del ítem según las reglas de orden es utilizada para permitir o impedir el acceso al ítem durante la navegación del usuario. Mientras que la

accesibilidad del ítem según las reglas de conocimiento es utilizada para determinar si tras la visita al ítem se ejecuta o no la regla de actualización asociada.

Cada vez que el usuario accede al contenido de un ítem, su modelo de usuario es actualizado, incrementando en uno el número de visitas a dicho ítem. Sin embargo, su estado de conocimiento sólo se actualiza si el usuario se encuentra preparado para asimilar el contenido del ítem visitado ya que, en otro caso, no es posible cuantificar el valor pedagógico de la lectura del ítem.

De esta manera, al igual que en el resto de modos (capítulo 15), la regla de actualización del ítem visitado sólo se aplica cuando el estado de conocimiento actual del usuario satisface las restricciones de accesibilidad de al menos una de las reglas de conocimiento asociadas.

En la figura 5 se muestra de forma esquemática el procedimiento seguido para determinar la posibilidad de visitar un ítem $c_k.i_j$, y en caso de producirse, actualizar las visitas del usuario, atributo *visitas(i_j)*, y si tiene lugar también su estado de conocimiento, atributo *K(i_s)* de los ítems incluidos en el cuerpo de la regla $Ru(i_j)$.

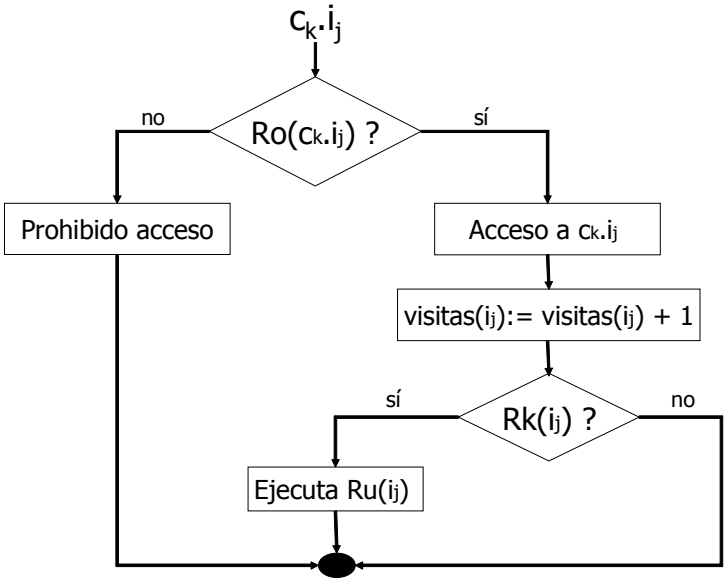


Figura 5: Acceso y actualización para un ítem $c_k.i_j$

6. CONCLUSIONES

El modo de navegación por relación conceptual permite al usuario recorrer una determinada parcela de conocimiento de una forma flexible, pero fiel con las relaciones existentes entre los conceptos. Esto es, cuando el usuario termina de leer el contenido de un ítem y pasa a otro, se asegura que la nueva información está directamente relacionada con la anterior, lo cuál supone un punto muy favorable en la compresión del material estudiado.

Como conclusión podemos caracterizar la navegación por relación conceptual a través de una serie de elementos, que podemos clasificar, según su utilidad, en tres categorías:

- Elementos estructurales destinados a **restringir la navegación** del usuario: relaciones conceptuales, restricciones de navegabilidad y reglas de orden.

    - Las restricciones de navegabilidad trasforman las relaciones conceptuales de la estructura conceptual de presentación en enlaces de navegación.

    - Las reglas de orden exigen el seguimiento de estos enlaces de navegación e imponen nuevas restricciones para la visita a un ítem. De forma que, finalmente, establecen un orden parcial en el proceso de navegación del usuario.

- Los elementos implicados en el proceso de **actualización del modelo de usuario**: reglas de conocimiento y reglas de actualización.

    - Cuando el usuario visita un ítem, además de contabilizar la nueva visita en el modelo de usuario, el sistema evalúa la accesibilidad del ítem de acuerdo a sus reglas de conocimiento y

    - , en caso afirmativo actualiza el conocimiento del usuario mediante la ejecución de su regla de actualización.

- Las **técnicas de adaptación** realizadas sobre la estructura de navegación: anotación y deshabilitación.

    - Resaltar el concepto actual con un borde más grueso  .

    - Resaltar las relaciones conceptuales que parten del concepto actual con una línea más gruesa .

    - Resaltar positivamente los conceptos disponibles con el fondo  .

    - Resaltar negativamente con el fondo  los ítems que aún estando ligados a un concepto disponible no pueden ser visitados.

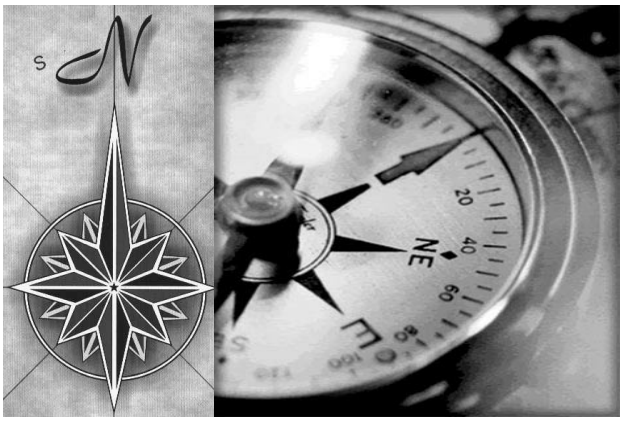
    - Resaltar positivamente los ítems que se pueden visitar con el fondo  .

    - Deshabilitar la funcionalidad de los ítems que no pueden ser visitados.

    - Anotar junto a cada ítem el número de veces que ha sido visitado  .

CAPÍTULO 18

Navegación por Conocimiento



Resumen

En este capítulo se describe un modo de navegación donde prima el estado de conocimiento del usuario, sus intereses, metas y preferencias. Se justifica la necesidad de este nuevo modo de navegación restringida, poniendo de manifiesto sus principales diferencias con la navegación por relación conceptual. Después, se establecen las restricciones y procedimientos utilizados para determinar cuándo un ítem es accesible, y cuándo es, además, idóneo. Se especifican los métodos de adaptación aplicados, poniendo de manifiesto la gran capacidad del sistema para adaptarse al usuario durante este modo de navegación. Y, para concluir se describe la actualización llevada a cabo sobre el modelo del usuario a media que éste navega.

Tabla de contenidos

Navegación por Conocimiento

1. INTRODUCCIÓN

En la navegación por conocimiento al igual que en la navegación por relación conceptual (capítulo 17), no siempre todos los ítems pueden ser visitados por el usuario. Esto es, existen **restricciones** que el usuario debe satisfacer para poder inspeccionar el contenido de un ítem. Ambos tipos de navegación se realizan en una red semántica que coincide con la **estructura conceptual de presentación** sobre la que se define la estructura de aprendizaje elegida específicamente para el usuario. Y, en ambos, la unidad de navegación es el **ítem**.

La diferencia entre los modos de navegación por relación conceptual y por conocimiento, radica precisamente en el tipo de restricciones que se imponen, en uno y otro, para el acceso a los ítems. En el primer caso se trata de restricciones de orden (capítulo 17) que controlan la navegación del usuario, decidiendo si éste puede o no visitar un ítem en función de los ítems que ha visitado anteriormente. Mientras que, en el segundo caso, como su propio nombre indica, se trata de **requisitos de conocimiento** (capítulo 8) que marcan la accesibilidad e idoneidad de un ítem, según el usuario conozca o no otros ítems necesarios y con qué grado.

> **Def 18.1 [Navegación por conocimiento]** La navegación por conocimiento es un modo de navegación restringido, que establece la accesibilidad e idoneidad de un ítem en función del estado de conocimiento del usuario y los requisitos pedagógicos impuestos por el autor para su visita. La unidad de navegación es el ítem y la red semántica proporcionada coincide con la EC_P^k incluida en la EC_A^j elegida para el usuario.

Como se ha visto anteriormente, en la navegación por relación conceptual el proceso de navegación del usuario debe ser coherente con las relaciones semánticas establecidas entre los conceptos. No es posible pasar de un ítem $c_k.i_j$ a otro $c_s.i_t$, si se asocian a conceptos diferentes ($c_k \neq c_s$) y entre ellos no existe una relación navegable en ese sentido, $r_c{:}c_k \rightarrow c_s$. Esto implica, con la salvedad de los *starting-points*, que la navegación por relación conceptual depende del concepto actual, es decir, del concepto al que se asocia el último ítem visitado.

Por el contrario, la navegación por conocimiento no se preocupa de los ítems que ha visitado el usuario, del orden en que esto ha sucedido, ni mucho menos del concepto actual y las relaciones semánticas navegables desde él. En este tipo de navegación, lo que realmente importa es el **estado de conocimiento del usuario** y si en dicho estado se satisfacen los requisitos de conocimiento impuestos por el autor para permitir la visita a un ítem y considerarla idónea.

En la navegación por relación conceptual se exige haber visitado $c_k.i_j$ antes de visitar $c_s.i_t$, mientras que en la navegación por conocimiento se exige que para visitar i_t el conocimiento sobre i_j sea superior a un determinado grado, siendo indiferente el modo en que éste se consigue:

a) puede ser visitando dicho ítem, i_j,

b) pero también visitando otros ítems que incrementen en su regla de actualización el conocimiento sobre i_j hasta el grado requerido.

Por lo tanto, en la navegación por conocimiento, aunque existe cierto orden entre los ítems, éste no suele ser estricto. Es decir, si se establece que es necesario conocer antes i_j que i_t, no significa, en todos los casos, que haya que visitar i_j antes que i_t. Sólo es así cuando la visita del ítem i_j es la única forma de adquirir conocimiento sobre él. En consecuencia, para lograr el conocimiento necesario para visitar un ítem, el usuario puede seguir diferentes procesos de navegación.

Cuando el conjunto de reglas de actualización coincide con el generado por defecto (capítulo 9), $\forall i_j$, *Ru(i_j): Si visitado(i_j) entonces Fix-abs(i_j, "total");*, la diferencia entre las reglas de orden y de conocimiento es que en éstas últimas no se distingue entre visita *anterior* y *previa*, puesto que no se tiene en cuenta el concepto actual ni las relaciones conceptuales a la hora de establecer el orden entre los items, sólo que items hay que conocer para poder conocer otros.

Las diferencias explicadas entre ambos modos de navegación, potencian la navegación por relación conceptual cuando se desea hacer hincapié en las relaciones semánticas existentes entre los conceptos, y la navegación por conocimiento cuando lo que más preocupa es que el usuario asimile correctamente el contenido de los ítems que inspecciona.

2. RESTRICCIONES DE NAVEGACIÓN

La navegación restringida por conocimiento ajusta el proceso de navegación del usuario a sus necesidades particulares. En concreto, el conjunto de ítems accesibles para el usuario, en un determinado paso de navegación, depende directamente del estado de conocimiento que actualmente posee. De modo que el usuario no puede visitar un ítem, si para asimilar correctamente su contenido, es necesario disponer de un nivel de conocimiento mayor que el suyo.

El sistema se encarga de identificar los ítems para cuya comprensión el usuario no está actualmente preparado, en función de lo establecido por el autor en las reglas de conocimiento (capítulo 8). Evaluando estas mismas reglas, el sistema también consigue reconocer los ítems que contienen información obvia o redundante para el usuario.

Mientras que en el primer caso (ítem no accesible) se prohíbe la visita del ítem, en el segundo (ítem no idóneo) simplemente se desaconseja. De esta forma, el sistema dirige al usuario en su recorrido por la estructura de navegación, impidiendo que lea información no inteligible para él, a la vez que le aparta voluntariamente de los ítems cuya lectura no le aportará nada nuevo.

Así pues, un ítem es **accesible** cuando el usuario tiene un conocimiento previo suficiente para entender y aprovechar la información que contiene. Y, además, diremos que es **idóneo** si éste incluye información inédita, novedosa o complementaria, que permite al usuario adquirir conocimiento nuevo o asentar y refinar el que ya tiene.

El procedimiento que sigue el sistema para determinar si un ítem es accesible e idóneo consiste en la evaluación de las reglas de conocimiento (Rk) incluidas en la estructura

de aprendizaje actual, EC_A^j = (EC_M, Rw, MU, EC_P^k, $RTnb^i$, Ro^i, Ru^j, Rk^j). A través de estas reglas, $Rk(EC_A^j)$, es el autor del sistema hipermedia, quien decide:

a) qué conocimiento previo es necesario para que un ítem sea accesible, y

b) cuando éste deja de ser idóneo.

Tal y como se especificó en el capítulo 8 (Reglas de conocimiento), cada ítem i_j tiene un conjunto de reglas de conocimiento asociadas $Rk(i_j) = \{Rk^1(i_j), Rk^2(i_j), ..., Rk^r(i_j)\}$. Cada una de estas reglas está compuesta por un antecedente de accesibilidad y otro de idoneidad, en la forma que sigue:

$Rk^i(i_j)$: antecedente-accesibilidad *and‴* antecedente-idoneidad → Visitar(i_j)

Cada antecedente se compone de cero o más restricciones, de accesibilidad o idoneidad según corresponda, unidas mediante el operador lógico clásico *and*.

En una **restricción de accesibilidad**, el autor exige un valor de conocimiento mínimo, *val1*, sobre un ítem i_k si, a su juicio, la información que presenta debe ser aprendida (en al menos ese grado) antes de acceder al ítem i_j. Esto es, para que i_j sea accesible, $K(i_k)$ debe ser mayor o igual que *val1*.

En una **restricción de idoneidad**, el autor establece el valor máximo de conocimiento, *val2*, sobre un ítem i_k para que siga teniendo sentido visitar i_j. Esto es, para que i_j sea idóneo, $K(i_k)$ debe ser menor o igual que *val2*.

Un ítem i_j es accesible e idóneo cuando se satisfacen ambos antecedentes en al menos una de sus reglas de conocimiento. Para que sea accesible basta con que se satisfaga el antecedente de accesibilidad de una de estas reglas.

3. ADAPTACIÓN DE LA ESTRUCTURA DE NAVEGACIÓN

La adaptación realizada en este modo de navegación puede ser dividida en tres grupos: ocultación de ítems inaccesibles y anotación de ítems no idóneos (3.1), anotación del estado de conocimiento del usuario (3.2) y adaptación a sus preferencias, intereses y metas (3.3).

3.1 Ocultación de ítems inaccesibles y anotación de ítems no idóneos

En cada paso de navegación, el sistema evalúa las reglas de conocimiento definidas en la estructura de aprendizaje actual, $Rk(EC_A^k)$, sobre el estado de conocimiento que en ese momento posee el usuario y que se encuentra almacenado en su modelo. De esta forma, consigue averiguar el carácter accesible e idóneo de cada uno de los ítems de la estructura que actualmente recorre el usuario.

Después, el sistema integra la información obtenida sobre la propia estructura de navegación. Se trata de una integración tanto visual como funcional. Mediante la primera se hace partícipe al usuario de la accesibilidad e idoneidad de un ítem a través de un simple vistazo. Mediante la segunda se impide la visita del usuario a los ítems identificados como no accesibles.

Para llevar a cabo ambas tareas se aplican las técnicas de adaptación que se describen brevemente a continuación:

- **Ocultación y deshabilitación de los ítems no accesibles:**

Los ítems identificados como no accesibles deben ser "escondidos" para disuadir al usuario de seleccionarlos. Para ello, el borde y el nombre del ítem se dibujan en color gris claro, de modo que apenas sean perceptibles. Con objeto de asegurar que el usuario no accede al contenido del ítem, la funcionalidad del enlace se deshabilita, de modo que aunque se haga *clic* sobre él no se muestra la información de destino.

El estado de conocimiento del usuario aumenta a medida que éste navega en la estructura proporcionada. Por lo tanto, cuando alcanza el conocimiento mínimo requerido para acceder a un ítem, dicho ítem se va a mantener accesible de ahí en adelante.

Esto es, supongamos que en un momento determinado, el usuario satisface todos los requisitos de accesibilidad, $K(i_k) \geq val1$, de una regla de conocimiento $Rk^i(i_j)$. Es evidente, que después de visitar otro ítem cualquiera, su nuevo estado de conocimiento va a ser igual o superior al que tenía previamente a la visita, y por lo tanto, va a seguir satisfaciendo todos los requisitos impuestos en $Rk^i(i_j)$ para considerar i_j accesible.

El hecho de que las reglas de actualización garanticen que el usuario no pierde conocimiento tras la lectura de un ítem (en todo caso lo gana), permite agilizar el proceso de evaluación que realiza el sistema para determinar la accesibilidad de los ítems. Ya que no tiene que reconsiderar la accesibilidad de los ítems que actualmente lo son, simplemente seguir considerándolos accesibles[1].

- **Anotación de los ítems no idóneos:**

Aquellos ítems que siendo accesibles no son idóneos se marcan de forma especial para que el usuario conciba su visita como permitida pero inadecuada. Concretamente, el borde del ítem y su asociación funcional se dibujan con una línea discontinua. En este caso, no se deshabilita el enlace, de forma que se deja a la discreción del usuario visitar o no el ítem, siendo consciente de que esa visita se desaconseja.

Puesto que el conocimiento del usuario crece o se mantiene tras las visitas que realiza, un ítem puede ser idóneo en un momento y dejar de serlo en el siguiente. Pero como caso excepcional, si un ítem se hace idóneo a través de una regla de conocimiento donde el antecedente de idoneidad es *true*, el sistema puede seguir considerándolo idóneo de ahí en adelante, sin necesidad de reevaluar sus reglas.

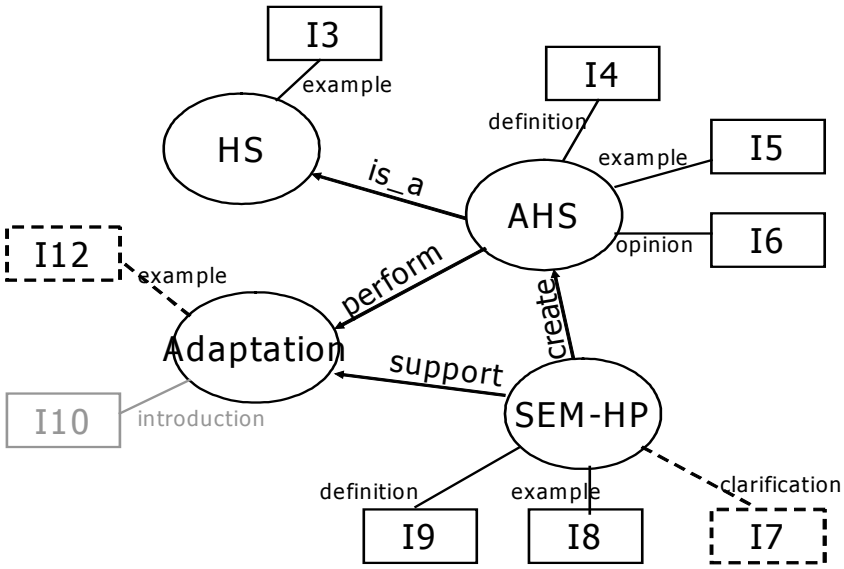
Por otro lado, si en un determinado momento no se satisface un requisito de idoneidad $K(i_k) \leq val2$ en una regla de conocimiento $Rk^i(i_j)$, en el momento siguiente tampoco se puede satisfacer. Por este motivo, cuando un ítem i_j deja de ser idóneo de acuerdo a una

[1] Exceptuando, claro está, el caso en que el usuario disminuye su estado de conocimiento ejerciendo un control explícito sobre su modelo de usuario, o cambia de EC_A.

regla $Rk^i(i_j)$, el sistema no tiene que volver a evaluarla para determinar, más adelante, la idoneidad del ítem.

De este modo, la idoneidad sólo se evalúa para los ítems accesibles y sobre las reglas no descartadas previamente. De manera que, únicamente, cuando un ítem deja de ser idóneo de acuerdo a todas sus reglas de conocimiento, el sistema deja de reconsiderar su idoneidad, desaconsejando su visita de ahí en adelante (ver nota al pie 1).

En el ejemplo de la figura 1 vemos la anotación realizada sobre una estructura de navegación a partir del estado de conocimiento del usuario y las restricciones de conocimiento que se muestran debajo.



Conocimiento del usuario

K(I3)	3
K(I4)	2
K(I5)	1
K(I6)	0
K(I7)	3
K(I8)	4
K(I9)	0
K(I10)	1
K(I12)	4

Reglas de Conocimiento

I3	Rk(I3) = ∅
I4	Rk(I4) = ∅
I5	Rk^1(I5): K(I3)≥2 and‴ K(I4)≤3 → Visitar(I5)
I6	Rk^1(I6): K(I5)≥3 and‴ true → Visitar(I6) Rk^2(I6): K(I4)≥2 and‴ true → Visitar(I6)
I7	Rk^1(I7): true and‴ [K(I5)≤2 and K(I8)≤2] → Visitar(I7)
I8	Rk^1(I8): K(I3)≥2 and‴ true → Visitar(I8)
I9	Rk(I9) = ∅
I10	Rk^1(I10): [K(I3)≥4 and K(I4)≥3] and‴ true → Visitar(I10)
I12	Rk^1(I12): K(I3)≥3 and‴ K(I8)≤3 → Visitar(I12)

Figura 1: Anotación de ítems accesibles e idóneos

Observe que el ítem I10 aparece oculto para evitar que sea accedido por el usuario. Esto se debe a que no se satisface el antecedente de accesibilidad de su única regla de

conocimiento. Esto es, el grado de conocimiento del usuario sobre I4 e I3 es menor al exigido en la regla Rk^1(I10). Además, la funcionalidad del enlace I10 ha sido deshabilitada. Por lo tanto, aunque el usuario insista en seleccionar dicho ítem, el navegador no visualizará su contenido.

Los ítems I7 e I12 aparecen con línea discontinua indicando que son accesibles pero no idóneos. Esto se debe a que, en ambos casos, el usuario supera el conocimiento máximo permitido sobre I8 para que la visita a dichos ítems sea aconsejable. No obstante, los enlaces conservan su funcionalidad. De manera, que en caso de ser seleccionados, el sistema pone a disposición del usuario la información contenida en éstos.

El resto de los ítems son accesibles e idóneos; bien porque no tienen ninguna restricción asociada como I3, I4 e I9, o bien porque se cumplen todos los requisitos existentes en al menos una de sus reglas de conocimiento. Observe que para I6 basta con que se satisfaga la segunda de sus reglas, Rk^2(I6).

Adviértase, también, que tras la siguiente visita del usuario, el sistema sólo tiene que reconsiderar la accesibilidad del ítem I10, ya que el resto va a seguir siendo accesible. Además, puesto que los ítems I7 e I12 tienen una única regla de conocimiento asociada y según ésta no son idóneos, tampoco van a serlo en el futuro. Del resto de los ítems, sólo es necesario reevaluar la idoneidad del ítem I5, los otros van a seguir siendo idóneos puesto que su antecedente de idoneidad es *true*.

3.2 Anotación del estado de conocimiento del usuario

De acuerdo a este tipo de navegación, un ítem puede ser visitado o no por el usuario, dependiendo del estado de conocimiento que posee en ese determinado momento. Es pues, el autor quién, a través de las reglas de conocimiento, encauza la navegación del usuario, apartándole de los ítems excesivamente complicados hasta que éste capacitado para entenderlos y desaconsejándole la visita de los ítems cuya información ya domina.

Dado que el estado de conocimiento del usuario limita y condiciona sus posibilidades de elección, consideramos que debe estar disponible para éste. No obstante, el usuario puede consultar su estado de conocimiento en su modelo de usuario, tantas veces como considere oportuno (capítulo 12). Sin embargo, al igual que en la navegación por relación conceptual se anota sobre la propia red semántica la información utilizada para restringir su navegación. En este caso, también es conveniente *anotar sobre la estructura de navegación el estado de conocimiento actual del usuario* que la utiliza. De modo, que con un simple vistazo y sin necesidad de abrir el modelo de usuario, éste pueda percatarse de cuál es su situación actual.

Representar el conocimiento del usuario directamente sobre la estructura de navegación tiene un claro provecho para el usuario. Por un lado, hace a éste **consciente de su proceso de aprendizaje**, ya que puede observar sin esfuerzo, cómo a medida que visita ítems en el sistema, su estado de conocimiento aumenta. Por otro lado, el hecho de saber su grado de conocimiento sobre cada ítem, permite al usuario ser más reflexivo en su proceso de navegación. Por ejemplo, mitiga que inspeccione ítems caprichosamente y fomenta que recorra la información con la intención de aumentar su estado de conocimiento y avanzar en el aprendizaje.

El proceso seguido para representar el estado de conocimiento del usuario sobre la estructura de navegación consiste en adjuntar junto a cada ítem i_j, el grado de conocimiento que sobre él posee ahora el usuario, **$K(i_j)$**. Esta información, $K(i_j)$, $\forall i_j \in I(EC_P^k)$, es automáticamente extraída del modelo del usuario. Puesto que el modelo se actualiza después de cada visita del usuario, la anotación realizada sobre la estructura de navegación se renueva después de cada paso.

En el mismo modo, se registra el grado de conocimiento que el usuario posee sobre cada concepto. De nuevo, el sistema obtiene esta información consultando en el modelo del usuario el valor de conocimiento, **$K(c_i)$**, para cada concepto c_i incluido en la estructura de navegación actual, esto es $\forall c_i \in C(EC_P^k)$.

En ambos casos, el grado de conocimiento es expresado mediante una etiqueta semántica: "nulo", "bajo", "medio", "alto" o "total", y no con el valor numérico correspondiente. De esta forma es más claro e intuitivo para el usuario. En el caso de un ítem, la transformación del valor numérico de $K(i_j)$ a una etiqueta semántica es directa, ya que dicho valor coincide exactamente con 0, 1, 2, 3 o 4 (capítulo 8).

Sin embargo, en el caso de un concepto, el grado de conocimiento $K(c_i)$ se calcula mediante la ejecución de su regla de peso, $Rw(c_i)$, la cual es una función matemática que puede devolver un valor real situado entre dos de los anteriores valores enteros. En este caso, la etiqueta semántica se obtiene tras una aproximación del valor real obtenido al valor entero más próximo. Y, la anotación de dicho grado de conocimiento debe estar precedida del símbolo ≈, para indicar que no es exacto (capítulo 11, Reglas de peso y capítulo 12, Modelo de usuario).

Con el objetivo de representar, lo más fielmente posible, el conocimiento acerca de un concepto, se puede acompañar el símbolo de aproximación, ≈, con una flecha hacia arriba, ↑, o hacia abajo, ↓, para indicar si el conocimiento real del usuario acerca del concepto es superior (redondeo por defecto) o inferior (redondeo por exceso) al grado de conocimiento anotado.

Por ejemplo, si el conocimiento obtenido para un concepto c_i ejecutando su regla de peso $Rw(c_i)$, es $K(c_i) = 2{,}35$, el redondeo le asigna la etiqueta semántica "medio" que corresponde al valor entero 2. En este caso la anotación debe ser **≈↑** medio, indicando que el conocimiento del usuario acerca de c_i está muy próximo a "medio" y que dicha proximidad es por exceso.

Si por el contrario el conocimiento acerca del concepto es 1,70, $K(c_i) = 1{,}70$, el redondeo asigna, también, la etiqueta "medio", pero en este caso el redondeo ha sido a la alza. Por lo tanto, la anotación debe ser **≈↓** medio, de forma que el usuario sepa que aunque su conocimiento acerca de c_i está próximo a "medio" aún no alcanza dicho valor.

Además del estado de conocimiento del usuario, es conveniente anotar sobre la estructura de navegación cierta información que sirva de soporte para la orientación. En este caso no tiene sentido mostrar el número total de veces que el usuario ha visitado cada ítem, como en la navegación por relación conceptual.

Únicamente se indica si el ítem ha sido visitado alguna vez (visitas(i_j)>0) o no (visitas(i_j)=0) por el usuario. Para ello, el sistema consulta el modelo del usuario, comprobando el valor del atributo **visitas(i_j)**, y colorea en oscuro el interior de los rectángulos que representan ítems previamente visitados. Esta anotación de los enlaces visitados es típica en los sistemas web tradicionales, y es muy útil para evitar que el usuario visite repetidamente un ítem de forma no intencionada.

En la figura 2 se muestra una posible anotación de conocimiento realizada sobre la estructura de navegación presentada en la figura 1.

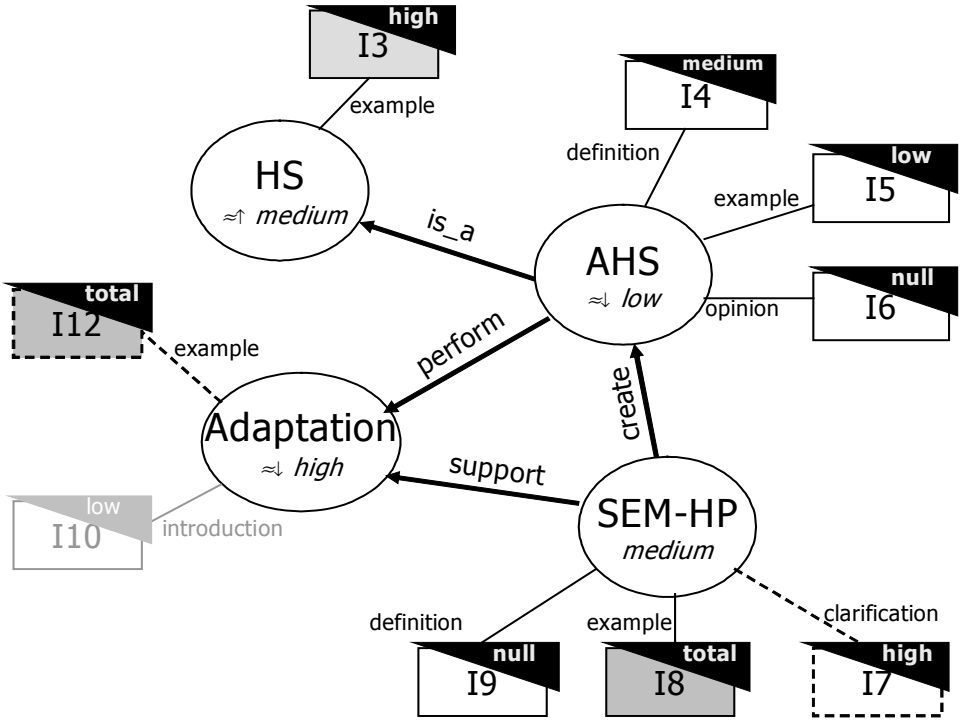


Figura 2: Anotación de conocimiento

Observe que para adjuntar el grado de conocimiento junto a cada ítem se ha utilizado un triángulo negro en la esquina superior derecha. Mientras que el conocimiento acerca de un concepto se muestra debajo del nombre de éste, dentro del círculo que lo representa.

Adviértase, además, que los únicos ítems visitados por el usuario en el ejemplo son I3, I8 e I12.

Una vez que el estado de conocimiento del usuario se refleja en la estructura de navegación, éste puede *adivinar las relaciones de actualización* existentes entre los ítems disponibles. Por ejemplo, si visita un ítem i_j y observa que como consecuencia se ha incrementado el conocimiento anotado sobre i_j e i_k, sabe que de alguna manera el contenido de i_k está relacionado con el de i_j, ya que al leer este último ha aprendido también algo sobre el primero.

Además, el usuario puede *intuir los requisitos de conocimiento* existentes entre los ítems de la estructura actual. Por ejemplo, si en un determinado momento un ítem i_r no es accesible y tras visitar i_j e incrementarse el conocimiento sobre i_j e i_k, el ítem i_r aparece ahora como accesible, es porque asociado a éste existe un requisito de

accesibilidad, recientemente satisfecho, sobre i_j y/o i_k. Del mismo modo, si al incrementarse el conocimiento sobre i_j un ítem i_r deja de ser idóneo, es porque existe asociado a i_r una restricción de idoneidad que exige un conocimiento máximo sobre i_j que acaba de ser superado.

Respecto a los conceptos, disponer del grado de conocimiento es muy útil ya que desde un punto de vista pedagógico suele ser más conveniente *evaluar el conocimiento* acerca de los conceptos que sobre trozos de información específicos.

Observando el ejemplo de la figura 2, podemos ver: que el grado con el que el usuario conoce la definición de *AHS* incluida en I4 es "medio", que su conocimiento acerca de los ejemplos de *AHSs* mostrados en I5 es "bajo", y que desconoce completamente la opinión ofrecida en I6 acerca de éstos.

A partir de esta información, el usuario podría hacerse una idea de cuánto conoce el concepto *AHS*. Sin embargo, se trataría de una idea inexacta, puesto que no sabe cuan importante es el conocimiento de cada uno de esos ítems para conocer el concepto, ni si existen más ítems asociados a dicho concepto en otras estructuras.

Por ese motivo, el conocimiento del usuario acerca del concepto *AHS*, calculado a partir de su conocimiento sobre todos los ítems que tratan sobre él y el peso asociado por el autor a cada uno, es anotado junto al concepto. De esta forma, el usuario puede ver que, concretamente, su conocimiento sobre dicho concepto está entre "nulo" y "bajo", más próximo a éste último valor ($\approx\downarrow$low).

El conocimiento de un concepto depende en mayor o menor medida del nivel con que el usuario conoce cada uno de los ítems ligados a él en la EC_M. Así pues, a veces, aún consiguiendo un conocimiento "total" sobre cada uno de los ítems ligados a un concepto en la estructura actual, EC_A^j, no se obtiene un conocimiento "total" sobre el concepto.

Anotando el conocimiento, también sobre los conceptos, se permite al usuario observar esta circunstancia. Haciéndole consciente de que la navegación de la estructura actual no le va a permitir conocer plenamente el concepto. Por lo que debe navegar en otras estructuras, si quiere tener acceso a todos los ítems que tratan ese concepto y poder conocerlo totalmente.

3.3 Adaptación a las preferencias, intereses y meta del usuario

Teniendo en cuenta el estado de conocimiento del usuario, el sistema es capaz de adaptarse a las preferencias, intereses y metas de éste. Esta facultad constituye una inestimable ayuda para el usuario, ya que le facilita la posibilidad de realizar una navegación dirigida a sus objetivos y adaptada a sus preferencias. Para ello, el sistema aplica dos métodos diferentes: **anotación de ítems deseables** (capítulo 19) y **generación de rutas guiadas** (capítulo 20).

Ambos mecanismos, satisfacen la finalidad de orientar al usuario para conquistar el estado de conocimiento deseado. Sin embargo, mientras que el primero tiene principalmente carácter de recomendación: *¿Qué ítems puedo visitar a continuación para acercarme a mi meta?* El segundo constituye un camino directo a la meta, que

debe ser seguido en el orden especificado: *¿Qué ítem tengo que visitar en cada paso para lograr la meta?*

Los procedimientos particulares necesarios en cada método de adaptación son explicados con amplio detalle en los dos capítulos siguientes. Por lo que, en el presente capítulo simplemente se hace una introducción a los mismos.

3.3.1 Anotación de ítems deseables

De acuerdo a un conjunto de **intereses** Ints = $\{i_1,..., i_n, c_1,..., c_r\}$ o a una **meta de conocimiento** M = $\{(i_1, et_{SEM}^{1}),..., (i_n, et_{SEM}^{n}), (c_1, et_{SEM}^{1}),..., (c_r, et_{SEM}^{r})\}$, el sistema identifica y resalta sobre la estructura de navegación aquellos ítems cuya visita permite al usuario acercarse a la consecución de sus deseos: poder visitar los ítems marcados como interesantes en el primer caso, o lograr un estado de conocimiento donde se satisfacen todas las submetas impuestas en el segundo.

Los **ítems** resaltados en cada paso de navegación se califican como **deseables**, ya que **su visita tiene un efecto positivo**, en mayor o menor grado, para la satisfacción de los objetivos perseguidos por el usuario. Obviamente lo ítems marcados como deseables, dependen del objetivo del usuario, esto es de la meta de conocimiento y/o del conjunto de intereses específico. Pero también, del estado de conocimiento del usuario y las reglas de conocimiento que rigen su navegación en la estructura actual.

La idea es muy sencilla. Supongamos que el usuario ha marcado como interesante o meta un ítem i_j (Def 12.4, 12.7). En la navegación tradicional no tendría problema en seleccionarlo e inspeccionar su contenido en busca de aquella información que le interesa. Sin embargo, en la navegación por conocimiento es posible que ese ítem no sea actualmente accesible para él. En ese caso, el usuario reflexionaría en el siguiente modo: "*No puedo visitar el ítem que me interesa porque aún no estoy preparado para comprender el contenido del mismo. Bien, ¿cómo puedo adquirir esa preparación?, esto es* ***¿qué puedo hacer para que dicho ítem se haga accesible?***".

Es el autor quien decide qué conocimiento es necesario para asimilar correctamente el contenido de un ítem, plasmando esta decisión en forma de restricciones de accesibilidad en las reglas de conocimiento. Si el usuario tuviese a su disposición esa información, $Rk(i_j)$ en este caso, seguramente continuaría su reflexión en la siguiente forma: "*Bueno, si para acceder a* i_j *es necesario superar un determinado nivel de conocimiento sobre una serie de ítems* $i_1,..., i_k$*, voy a ver para cuales de estos ítems mi grado de conocimiento es inferior al exigido y a visitar cada uno de ellos*".

El argumento de nuestro perspicaz usuario es correcto: visitando, una o más veces, los ítems sobre los que existe una restricción de accesibilidad no satisfecha en $Rk(i_j)$, se consigue finalmente que el ítem i_j se vuelva accesible. Pudiendo, por tanto, leer la información que éste contiene, en posesión de un estado de conocimiento adecuado para extraer de esa lectura el máximo rendimiento posible. Sin embargo, las dificultades que se puede encontrar el usuario al seguir el anterior razonamiento conseguirían, en la mayor parte de los casos, malograr su empeño y hacerle desistir.

Dicha **dificultad** radica en la posibilidad, nada despreciable, de que uno o varios de esos ítems que pretende visitar para poder visitar después i_j no sean, tampoco, accesibles en ese momento. Entonces, el usuario debe repetir el razonamiento para cada uno de

ellos, de forma *recursiva* hasta que todos los ítems que necesita visitar le sean accesibles. Esto unido al hecho de que pueden existir *varias alternativas* para hacer accesible un ítem, es decir varias reglas de conocimiento asociadas a éste, complica el argumento del usuario acerca de: "¿Qué ítems tengo que visitar para poder acceder a i_j?" Una complejidad que se multiplica cuando pretende responder a esta pregunta no para un único ítem sino para un conjunto de éstos.

A través de la anotación de los ítems deseables, el sistema sigue por el usuario el razonamiento explicado, de modo que éste se despreocupa y tan sólo necesita saber que *al visitar un ítem marcado como deseable está contribuyendo a lograr el conocimiento necesario para satisfacer su meta y/o para convertir en accesible un ítem interesante.*

Dependiendo del ítem que visita el usuario y cuál es la repercusión de dicha visita en su estado de conocimiento, el sistema actualiza para el siguiente paso el conjunto de ítems deseables. En dicha **actualización** algunos **ítems deseables** pueden dejar de serlo y otros que no lo eran pasar a serlo ahora. Por ejemplo, si el ítem i_k era deseable únicamente para hacer accesible el ítem i_j, y con el nuevo estado de conocimiento, i_j se ha vuelto accesible, el ítem i_k dejaría de formar parte del conjunto de ítems deseables e i_j podría entrar en dicho conjunto.

En la figura 3 se muestra la interfaz con la que el usuario interacciona para navegar por conocimiento, suponiendo que la meta del usuario es alcanzar un conocimiento "total" sobre los ítems I5 e I10. Esto es M = {(I5, "total"), (I10, "total")}.

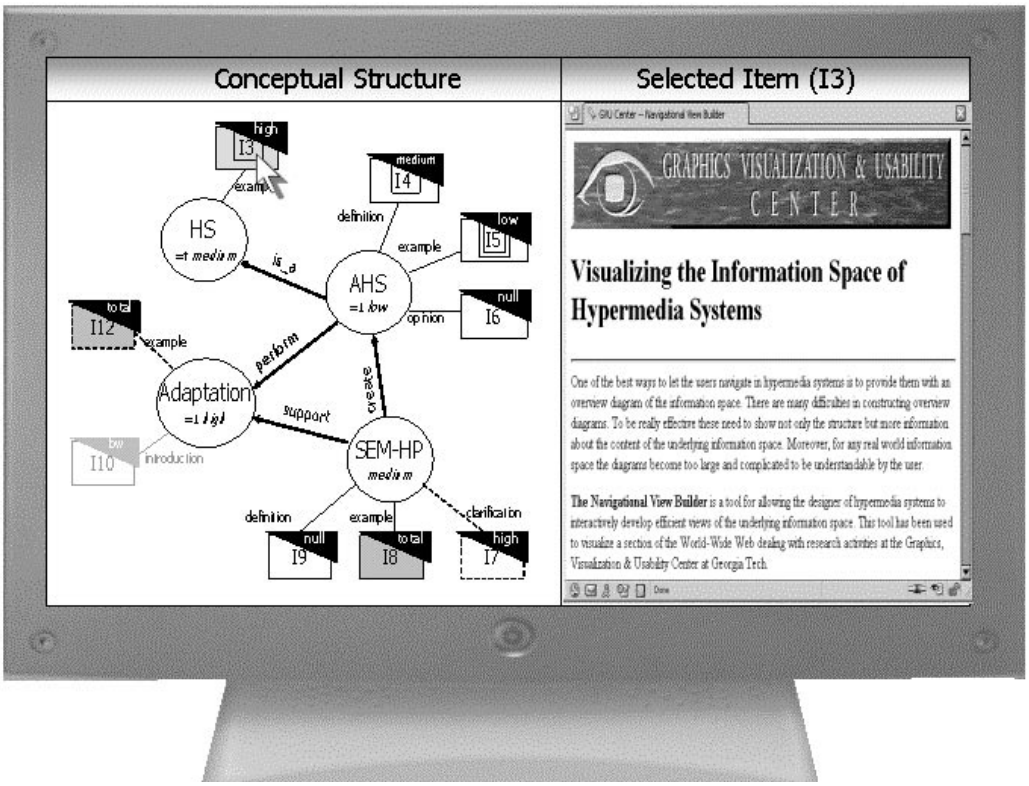


Figura 3: Anotación de ítems deseables

Observe que además de anotar el estado de conocimiento del usuario, ocultar y deshabilitar los ítems inaccesibles (I10), anotar los ítems no idóneos (I7, I12) y marcar

los ítems visitados (I3, I8, I12), se ha realizado una anotación de ítems deseables (I3, I4, I5).

Concretamente el ítem I5 aparece doblemente bordeado con un cuadrado indicando que es un ítem deseable y que, además, forma parte de la meta. Sin embargo, el ítem I10, también en la meta, no se marca como deseable, puesto que ni siquiera es accesible. En su lugar se señalan con un único cuadrado, para indicar que son ítems deseables pero no meta, los ítems I3 e I4. Ya que, de acuerdo a la regla de conocimiento Rk^1(I10) mostrada en la figura 1, la visita a estos ítems contribuye a que el ítem I10 se vuelva accesible.

En el ejemplo, haciendo caso de la sugerencia, el usuario ha seleccionado el ítem I3 cuyo contenido se muestra en el marco situado a la derecha.

En la anotación de ítems deseables se tienen en cuenta los intereses y la meta del usuario. Las preferencias del usuario, únicamente son utilizadas cuando el usuario solicita que el sistema obtenga el conjunto de ítems interesantes, o la meta, a partir de sus preferencias (capítulo 12, Modelo de Usuario). En este caso, el conjunto de ítems deseables permite asesorar al usuario para que visite los ítems cuyas características son más de su agrado.

3.3.2 Rutas guiadas personalizadas

Teniendo en cuenta las **preferencias del usuario** y su **meta de conocimiento** $M = \{(i_1, et_{SEM}^1),..., (i_n, et_{SEM}^n), (c_1, et_{SEM}^1),..., (c_r, et_{SEM}^r)\}$, el sistema es capaz de planificar una ruta que conduzca al usuario desde su estado de conocimiento actual hasta un estado meta. La **ruta** se define mediante un *conjunto de ítems y un orden secuencial* de visita. La longitud de la ruta, así como las características de los ítems incluidos en la misma va a depender, dentro de lo posible, de las preferencias expresadas al respecto por el usuario.

En este caso, se garantiza que si el usuario sigue la ruta, en el orden indicado y hasta el final, va a conseguir un estado de conocimiento donde se satisfacen cada una de las submetas expresadas en M. Para ello, el sistema partiendo del estado de conocimiento inicial del usuario realiza un **proceso de búsqueda**, basado en las reglas de conocimiento y de actualización definidas para la estructura actual.

Dicha búsqueda culmina con la identificación de una secuencia de ítems $i_1, i_2,..., i_s$, que siempre cumple las siguientes propiedades:

a) Los ítems de la ruta pueden ser visitados en el orden indicado. Esto es, primero i_1, luego i_2,..., y finalmente i_s.

Para ello es necesario que i_1 sea accesible en el estado de conocimiento inicial del usuario, que i_2 sea accesible en el estado alcanzado tras visitar i_1,..., y en general que i_j sea accesible en el estado alcanzado tras visitar en este orden: $i_1, i_2,..., i_{j-1}$.

b) Tras completar la ruta, el estado de conocimiento alcanzado por el usuario, s_f, satisface cada una de los requisitos de conocimiento impuestos en la meta, M.

Esto es, en s_f se cumple que $K(i_1) \geq$ valor-numérico(et_{SEM}^1),..., $K(i_n,) \geq$ valor-numérico(et_{SEM}^n), ..., $K(c_1) \geq$ valor-numérico(et_{SEM}^1), ..., $K(c_r) \geq$ valor-numérico(et_{SEM}^r).

El estado de conocimiento tras visitar un ítem, i_j, corresponde al resultado de aplicar $Ru(i_j)$ sobre el estado obtenido en el paso anterior. Así, el estado de conocimiento final s_f, se obtiene con la aplicación de las reglas de actualización $Ru(i_1)$, $Ru(i_2)$, ..., $Ru(i_s)$, en ese orden, a partir del estado inicial del usuario.

Pueden existir estados de conocimiento diferentes que, sin embargo, satisfacen una misma meta, y además, pueden existir varias rutas que conducen a un mismo estado, por lo tanto el sistema debe elegir una de entre todas las alternativas posibles. Esta elección, va a depender de la estrategia de búsqueda utilizada, pero también de la **calidad de la ruta** medida en función de las preferencias establecidas por el usuario.

De este modo, es necesaria una función que evalúe el costo de las rutas encontradas, con el objetivo de poder optar por la que presente el mejor valor. El costo de la ruta va a depender de la *longitud* de la misma y de la importancia que el usuario le concede a ésta, pero también del costo de cada uno de los ítems cuya visita se proyecta en la ruta. El *costo de un ítem* se calcula de modo que sea menor cuanto más se ajusten sus características a las que más gustan al usuario, y mayor en el caso opuesto.

La anotación de la ruta también puede hacerse sobre la estructura de navegación. Para ello, en el ejemplo de la figura 4, junto a cada ítem incluido en la ruta se muestra un "banderín" que indica el orden en que éste debe ser visitado.

Para generar la ruta se ha considerado: el estado de conocimiento que se visualiza en la propia estructura de navegación (figura 4), las reglas de conocimiento de la figura 1, las reglas de actualización que se muestran en la tabla 1 y la misma meta que en el ejemplo anterior, esto es, M = {(I5, "total"), (I10, "total")}.

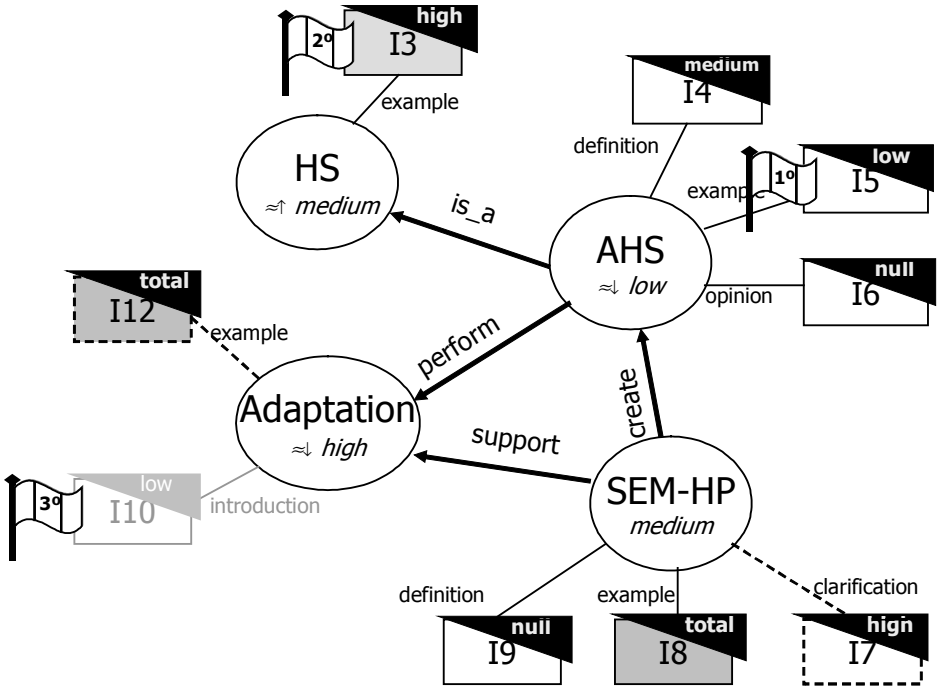


Figura 4: Mostrar la ruta guiada

Tabla 1: Reglas de actualización

Ru(I2)	Si Visitado(I2) entonces Fix-abs(I2, “total”);
Ru(I3)	Si Visitado(I3) entonces Fix-abs(I3, “total”), Inc-abs(I4, 1);
Ru(I4)	Si Visitado(I4) entonces Fix-abs(I4, “total”);
Ru(I5)	Si Visitado(I5) entonces Fix-abs(I5, “total”);
Ru(I6)	Si Visitado(I6) entonces Fix-abs(I6, “total”), Fix-abs(I7, “medio”);
Ru(I7)	Si Visitado(I7) entonces Inc-abs(I7, 2);
Ru(I8)	Si Visitado(I8) entonces Fix-abs(I8, “total”), Inc-rel(I9);
Ru(I9)	Si Visitado(I9) entonces Fix-abs(I9, “total”);
Ru(I10)	Si Visitado(I10) entonces Fix-abs(I10, “total”);
Ru(I12)	Si Visitado(I12) entonces Fix-abs(I12, “total”);

Como puede observar, la ruta generada se compone de tres pasos: primero visitar I5, luego visitar I3 y por último visitar I10. Al visitar I5 se alcanza conocimiento “total” sobre éste, superando así la submeta (I5, “total”). Para superar la submeta asociada a I10, es necesario visitar antes I3 cuya regla de actualización, tras ser ejecutada, sitúa al usuario en disposición de visitar I10, y obtener el conocimiento “total” requerido para éste en la meta (I10, “total”).

Note que no es necesario visitar I4, aunque sobre él existe una restricción de accesibilidad inicialmente no satisfecha en la regla de conocimiento de I10. Esto se debe a que la regla de actualización asociada a I3 incrementa el conocimiento del usuario acerca de I3 y, también, acerca de I4.

La generación de una ruta guiada supone dotar al usuario de una estrategia de navegación que le asegura la consecución de su meta de conocimiento, teniendo en cuenta el estado de conocimiento del que parte y sus preferencias respecto al tipo de ítems que desea visitar en el camino.

El sistema construye la estrategia de navegación analizando las distintas posibilidades que definen las reglas de conocimiento y actualización existentes, y optando por aquella que considera más cerca de las preferencias del usuario. Una vez obtenida la ruta, el usuario puede:

a) Seguir de manera ininterrumpida la ruta, de principio a fin, hasta superar el conocimiento deseado. En este caso, diremos que la navegación del usuario es totalmente guiada, ya que es el sistema quien dirige cada uno de sus pasos de navegación.

b) Intercalar entre los ítems de la ruta, la visita a algunos otros ítems accesibles que le apetezca visitar. En este caso, el usuario modifica la estrategia de navegación, mezclando la ruta generada por el sistema con sus propias decisiones.

En definitiva, al disponer de una ruta, el usuario tiene la tranquilidad de conocer un camino posible para llegar a su meta, pudiendo si lo desea, deleitarse durante el recorrido, inspeccionando algunos ítems que no pertenecen a la ruta. Para que en cualquier momento, el usuario conozca su posición en la ruta, el sistema elimina la anotación de los ítems ya visitados y mantiene la de los ítems que le faltan.

4. ACTUALIZACIÓN DEL MODELO DE USUARIO

Como en todos los tipos de navegación, también en éste el sistema actualiza el número de visitas realizadas por el usuario y el estado de conocimiento adquirido a través de éstas.

Respecto al primero no existe ninguna particularidad. Cada vez que el usuario inspecciona el contenido de un ítem i_j, el sistema incrementa en uno el contador de visitas asociado al ítem en su modelo de usuario. Esto es, **visitas(i_j)** = visitas(i_j) + 1.

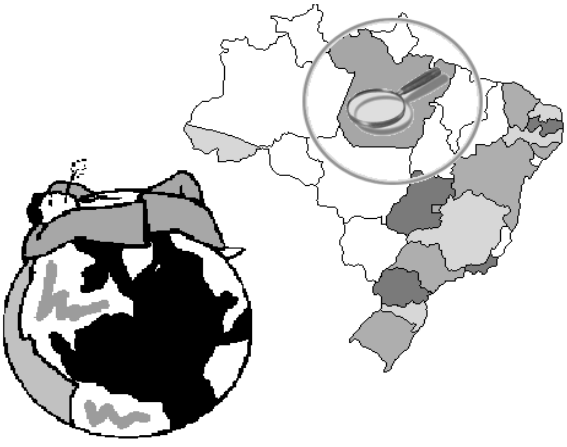
Respecto a la actualización de conocimiento, es aquí donde ésta cobra su máxima expresión. En la navegación por conocimiento, un ítem es accesible únicamente si el usuario está preparado para asimilar su contenido. Eso significa que el usuario no puede visitar un ítem si su estado de conocimiento no satisface todas las restricciones de accesibilidad en al menos una de las reglas de conocimiento asociadas al ítem. Por este motivo, siempre que el usuario visita un ítem i_j, el sistema ejecuta la regla de actualización correspondiente, Ru(i_j).

Dicho de otro modo, cada vez que el usuario visita un ítem, lo hace de acuerdo a las condiciones de acceso impuestas en sus reglas de conocimiento. Por tanto, después de leer el ítem, el grado de conocimiento del usuario sobre éste y posiblemente otros ítems relacionados puede ser incrementado. De manera que, muy a menudo, existe una relación "causa-efecto" entre la visita del usuario a un ítem y la actualización de su estado de conocimiento en el modelo de usuario.

Además, durante este modo de navegación, cabe esperar que el usuario interaccione más frecuentemente con su modelo para establecer metas de conocimiento, items interesantes, preferencias respecto al contenido de los items, respecto a la longitud de las rutas guiadas, etc.

CAPÍTULO 19

Ítems Deseables



Resumen

Este capítulo contiene una descripción detallada de la anotación de ítems deseables realizada durante la navegación por conocimiento. Para explicar esta técnica de adaptación, se definen las propiedades que tiene que satisfacer un ítem para que su visita sea deseable de acuerdo a los intereses del usuario, distinguiendo si se trata de un conjunto de ítems y/o conceptos interesantes o de una meta de conocimiento. Se especifican los procedimientos, que en cada caso sigue el sistema para identificar los ítems deseables, así como el proceso de anotación que tiene lugar sobre la estructura de navegación para dar a conocerlos. También se establece la forma en que el conjunto de ítems deseables se actualiza cuando cambian las circunstancias que lo generaron.

Tabla de contenidos

Ítems Deseables

1. INTRODUCCIÓN

Tal y como se describe en el capítulo 12, el modelo de usuario recoge cierta información sobre los intereses y deseos del usuario, con objeto de personalizar, en función de éstos, su proceso de navegación. Concretamente, los intereses que se contemplan en el modelo de usuario son: subdominio de interés, **ítems y conceptos interesantes** (Ints), y **meta de conocimiento** (M).

El subdominio de interés es consultado por el sistema para determinar, junto con la experiencia del usuario, qué estructura de aprendizaje le conviene más a éste (capítulo 14, Elección de la EC_A). De este modo, la estructura de navegación proporcionada captura, con el mayor grado posible, el subdominio de conocimiento en el cual el usuario está más interesado.

Sin embargo, dentro de un mismo subdominio, el usuario puede sentir más predilección por conocer unos ítems que otros. Este interés especial queda reflejado en el modelo de usuario a través del conjunto de ítems interesantes y la meta de conocimiento.

En el presente capítulo, se describe el procedimiento que el sistema aplica, para a partir de los ítems interesantes y/o la meta de conocimiento realizar un paso más en el proceso de adaptación al usuario. Este nuevo paso permite refinar la personalización de la estructura de navegación ofrecida al usuario, *resaltando sobre ésta aquellos ítems cuya visita aproxima al usuario a la consecución de sus metas e intereses*.

El motivo por el que esta técnica de adaptación es necesaria y útil es el siguiente: En la navegación por conocimiento el usuario no puede visitar cualquier ítem que se le antoje, sino sólo aquellos para cuya comprensión está preparado. Esto ocasiona, que a veces, estando interesado en inspeccionar la información de un ítem i_j no pueda hacerlo. La anotación de ítems deseables, va a orientarle sobre los ítems que puede visitar para conseguir que ese ítem que le interesa se vuelva accesible.

De este modo, durante la navegación del usuario, el sistema anota de forma especial aquellos ítems de la estructura de navegación que, actualmente son accesibles y dicho acceso puede ayudar al usuario a lograr sus intereses. El significado de "lograr sus intereses" depende del tipo de predilección, es decir, si se trata de un conjunto de ítems interesantes o de una meta de conocimiento. En el primer caso, para lograr los intereses basta con visitar cada uno de los ítems del conjunto, mientras que en el segundo, es necesario alcanzar un determinado grado de conocimiento sobre cada ítem incluido en la meta.

> **Def 19.1 [Ítem deseable]** Un ítem deseable es aquél cuya visita es provechosa desde la perspectiva de visitar los ítems interesantes o alcanzar la meta de conocimiento deseada.

El usuario percibe los ítems deseables a través de la anotación que el sistema realiza sobre la propia estructura de navegación, cuando ésta es recorrida en el modo de navegación por conocimiento (capítulo 18). Esta anotación simplemente pretende aconsejar al usuario acerca de qué ítems visitar, y en ningún caso es vinculante, es decir,

el usuario puede visitar o no los ítems deseables, y en caso de hacerlo, en el orden que él quiera.

El cálculo de los ítems deseables es similar para los dos tipos de intereses: ítems interesantes y meta de conocimiento. Para conseguir una exposición clara, el procedimiento establecido para averiguar y anotar los ítems deseables se ha dividido en dos partes. De manera que, primero se describe el proceso que tiene lugar para un conjunto de ítems interesantes (sección 2), y después sobre éste se explican las diferencias que existen cuando se trata de una meta de conocimiento (sección 3). Por último, en la sección 4, se indican las modificaciones necesarias cuando en el conjunto de intereses o en la meta de conocimiento aparecen involucrados también conceptos.

2. ÍTEMS DESEABLES PARA UN CONJUNTO DE ÍTEMS INTERESANTES

El primer paso, es definir formalmente qué condiciones o características debe cumplir un ítem para que el sistema lo identifique como deseable. Después, basándonos en esa definición, explicamos el algoritmo implementado para reconocer esos ítems con objeto de, finalmente, anotarlos sobre la estructura de navegación.

2.1 Definición de ítem deseable

Una vez que el usuario ha especificado un conjunto de ítems interesantes no vacío, Ints = $\{i_1, i_2,..., i_n\}$, y se encuentra navegando sobre una estructura $EC_A^j = (EC_M, Rw, MU, EC_P^k, RTnb^i, Ro^i, Ru^j, Rk^j)$, decimos que un ítem $i_j \in I(EC_P^k)$, es deseable si satisface dos propiedades:

Propiedad 1: El ítem i_j es un ítem **accesible**, es decir el estado de conocimiento del usuario satisface todas las restricciones de accesibilidad de al menos una de las reglas de conocimiento $Rk(i_j)$ asociadas a dicho ítem en la estructura de aprendizaje actual.

Propiedad 2.A: El ítem i_j es un ítem **interesante** para el usuario, es decir pertenece al conjunto de ítems marcados como interesantes en el modelo de usuario ($i_j \in$ Ints), o

Propiedad 2.B: El ítem i_j no es interesante, $i_j \notin$ Ints, pero una visita del usuario a i_j **contribuye** directa o indirectamente **a que un ítem interesante** i_f **se vuelva accesible**.

- La visita repetida a un ítem i_j *contribuye directamente a que el ítem i_f se vuelva accesible*, si existe, en alguna regla de conocimiento $Rk^i(i_f)$ asociada a i_f, una restricción de accesibilidad sobre i_j, $K(i_j) \geq$valor, actualmente no satisfecha por el usuario.

- La visita a un ítem i_j *contribuye indirectamente a que el ítem i_f se vuelva accesible*, si existe, en alguna regla de conocimiento asociada a i_f, una restricción de accesibilidad sobre un ítem i_t que aún no está satisfecha, y la visita repetida a i_j contribuye directa o indirectamente a que el ítem i_t se vuelva accesible.

En consecuencia, cuando el usuario visita un ítem deseable debe saber que está:

a) visitando un ítem que él mismo definió como interesante, o

b) visitando un ítem que le acerca al estado de conocimiento requerido para poder visitar uno o varios de los ítems interesantes.

Supongamos el conjunto de reglas de conocimiento que se muestran, en su representación interna, en la tabla 1, el conjunto de reglas de actualización por defecto mostradas en la tabla 2, y el estado de conocimiento inicial del usuario en la tabla 3.

Tabla 1: Reglas de conocimiento

Rk(I1)	Rk^1(I1): true *and‴* K(I4) ≤ 3 → Visitar(I1)
Rk(I2)	Rk^1(I2): K(I5) ≥ 3 *and‴* K(I3) ≤ 2 → Visitar(I2)
Rk(I3)	Rk^1(I3): [K(I1) ≥ 3 and K(I2) ≥ 1] *and‴* K(I1) ≤ 2 → Visitar(I3) Rk^2(I3): K(I4) ≥ 2 *and‴* true → Visitar(I3)

Tabla 2: Reglas de actualización

Ru(I1): Si Visitado(I1) entonces Fix-abs(I1, ”total”);
Ru(I2): Si Visitado(I2) entonces Fix-abs(I2, ”total”);
Ru(I3): Si Visitado(I3) entonces Fix-abs(I3, ”total”);
Ru(I4): Si Visitado(I4) entonces Fix-abs(I4, ”total”);
Ru(I5): Si Visitado(I5) entonces Fix-abs(I5, ”total”);

Tabla 3: Estado de conocimiento del usuario

K(I1)	K(I2)	K(I3)	K(I4)	K(I5)
0	0	0	0	0

Si el ítem I3 se hubiese marcado como interesante, el conjunto de ítems deseables sería {I1, I4, I5}. Los tres ítems son accesibles, a pesar del desconocimiento inicial del usuario, ya que I4 e I5 carecen de reglas de conocimiento asociadas y en la regla de I1 el antecedente de accesibilidad es *true*.

Las visitas de I1 e I4 contribuyen directamente a hacer accesible I3 ya que tras éstas se satisfacen las restricciones K(I1) ≥ 3 y K(I4) ≥ 2 impuestas en las reglas Rk^1(I3) y Rk^2(I3) respectivamente.

El ítem I2 tiene una restricción de accesibilidad no satisfecha en el cuerpo de Rk^1(I3), sin embargo, no es deseable porque actualmente no es accesible. Para que I2 sea accesible requiere que el conocimiento sobre I5 sea mayor o igual que 3. Por eso, se incluye en el conjunto de deseables el ítem I5 que, de este modo, contribuye indirectamente a la accesibilidad de I3.

Obviamente, cuando el usuario posee un estado de conocimiento que le permite acceder a todos los ítems interesantes, el conjunto de ítems deseables, al que notaremos **Dbls**, coincide exactamente con el conjunto de ítems interesantes, esto es Dbls = Ints.

En caso contrario, para calcular el conjunto de ítems deseables, el sistema construye un árbol con las *restricciones de conocimiento necesarias y no satisfechas para poder visitar los ítems interesantes*.

Def 19.2 [Tree$_D$] Denominamos Tree$_D$ al árbol que el sistema construye para, dado un conjunto de reglas de conocimiento, un conjunto de ítems interesantes y el estado actual del usuario, identificar el conjunto de ítems deseables. Los nodos del árbol se etiquetan con ítems y las ramas representan restricciones de accesibilidad no satisfechas sobre éstos. Además, la estructura del árbol es tal, que siempre los nodos de primer nivel se etiquetan con los ítems interesantes y los nodos del último nivel contienen los ítems deseables. Dbls = {nodo-hoja(Tree$_D$)}.

2.2 Construcción del Tree$_D$

De acuerdo a la definición anterior, el **árbol de ítems deseables** o Tree$_D$ es generado automáticamente por el sistema a partir de:

a) El conjunto de ítems interesantes especificado por el usuario, Ints.

b) El estado de conocimiento actual del usuario, estado$_K$.

c) El conjunto de reglas de conocimiento, Rk(EC_A^j), incluidas en la estructura de aprendizaje EC_A^j elegida para el usuario.

Tanto el conjunto de ítems interesantes como el estado de conocimiento actual del usuario son consultados por el sistema en el modelo del usuario. No obstante, no tendría sentido obtener los ítems deseables para un ítem interesante que no aparece en la estructura proporcionada para navegar al usuario. Por lo tanto, no se requieren todos los ítems interesantes, ni el estado de conocimiento completo, sino sólo la parte correspondiente al dominio de información mostrado en la estructura de navegación actual, esto es I(EC_P^k).

De este modo, del conjunto total de ítems interesantes, Ints, el sistema extrae únicamente aquellos que aparecen en la estructura EC_P^k sobre la que se define la EC_A^j elegida para el usuario. Este subconjunto, al que denominamos **Ints'**, va a ser el que se utilice para construir el Tree$_D$ y obtener los ítems deseables. Igualmente, el conocimiento del usuario sobre un ítem sólo es considerado si éste pertenece a I(EC_P^k) (véase la ecuación 1).

$$i_j \in \text{Ints'} \text{ si } \; i_j \in \text{Ints} \; \wedge \; i_j \in I(EC_P^k), \qquad (1)$$

$$K(i_j) \in \text{estado}_K\text{'} \text{ si } \; K(i_j) \in \text{estado}_K \; \wedge \; i_j \in I(EC_P^k)$$

Un ítem debe ser accesible para que se pueda catalogar como deseable, sin embargo, no es necesario que sea idóneo. Por este motivo, para calcular el conjunto de ítems deseables o lo que es lo mismo construir el Tree$_D$, solamente se evalúa el antecedente de accesibilidad de las reglas de conocimiento implicadas, Rk(EC_A^j), ignorando pues las restricciones de idoneidad presentes en éstas.

Tal y como se expuso en el capítulo 8 (Reglas de conocimiento), a partir de las reglas de conocimiento definidas por el autor, el sistema construye para cada ítem i_j un árbol de accesibilidad denominado **Tree$_K(i_j)$** que integra todas las restricciones de accesibilidad del conjunto de reglas de conocimiento $Rk^1(i_j)$, $Rk^2(i_j)$,..., $Rk^n(i_j)$ asociadas internamente a dicho ítem.

Así pues, para mejorar la eficiencia en el proceso de generación del árbol de ítems deseables ($Tree_D$), el sistema emplea los árboles de accesibilidad ($Tree_K$) construidos previamente para cada ítem involucrado, en lugar de su representación lógica equivalente en las reglas de conocimiento.

Para describir más claramente el conjunto de pasos que el sistema sigue a la hora de generar un $Tree_D$ nos serviremos del ejemplo que se expone a continuación.

En la figura 1 se muestra la estructura conceptual de presentación proporcionada al usuario para navegar. En ella, existen cuatro conceptos: *Loto*, *Asanas*, *Ha* y *Hatha Yoga*, y siete ítems identificados secuencialmente desde I1 hasta I7.

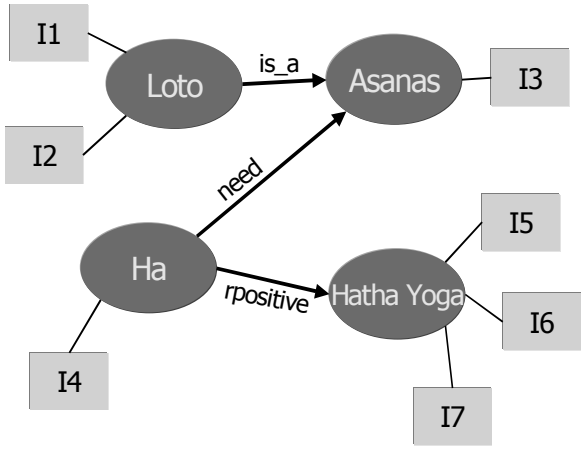


Figura 1: Ejemplo para la construcción del $Tree_D$

La figura 2 muestra la información necesaria para construir el árbol de ítems deseables para un usuario que recorre la estructura anterior. Dicha información incluye: ítems interesantes, estado de conocimiento y reglas de conocimiento.

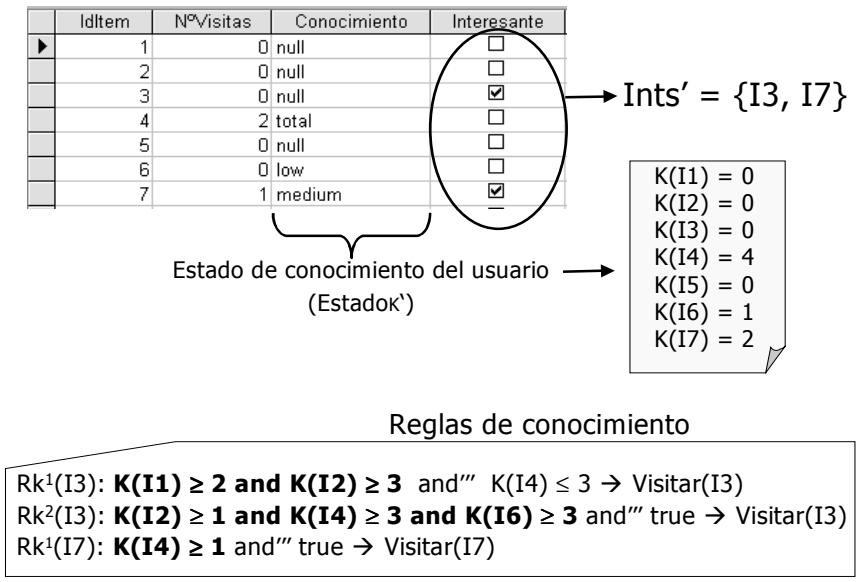


Figura 2: Información para obtener el $Tree_D$

En el ejemplo se ha supuesto que únicamente existen tres reglas de conocimiento definidas sobre la estructura de la figura 1. Dos de estas reglas se asocian al ítem I3 y la otra al ítem I7. Por lo tanto, el resto de ítems son accesibles sea cual sea el conocimiento del usuario.

Asimismo, se muestra la parte del modelo de usuario que contiene la información relativa a los ítems de la estructura de navegación actual. Advierta, como el sistema extrae de ésta, el estado de conocimiento actual del usuario, $estado_K$', y los ítems que le interesan especialmente, Ints'.

La **construcción del árbol de ítems deseables** comienza con la generación de un **nodo raíz** etiquetado como $Tree_D$. El **primer nivel** del árbol, es decir el que cuelga del nodo raíz, está formado por tantos nodos como ítems interesantes existen en el conjunto Ints'. Todos los nodos del primer nivel se unen mediante el operador *and*, indicando que para satisfacer el conjunto de intereses es necesario satisfacer cada uno de ellos (figura 3.1).

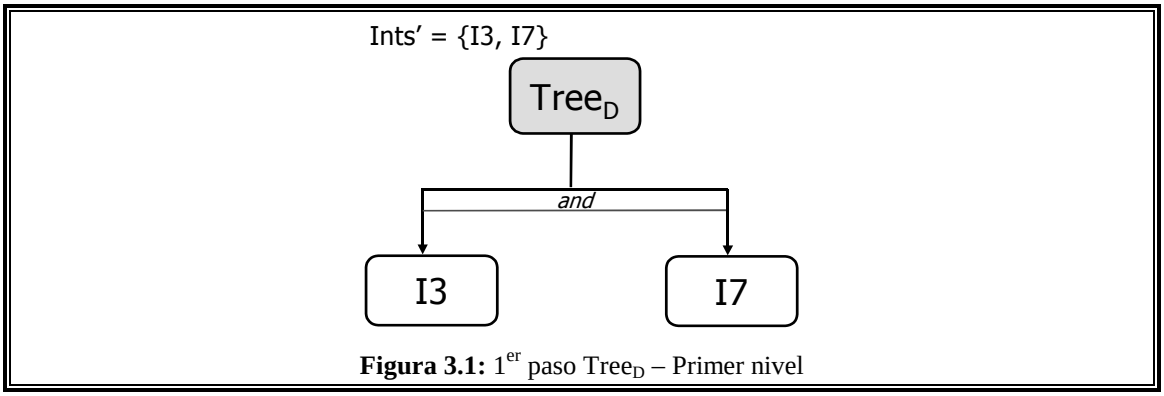


Figura 3.1: 1er paso $Tree_D$ – Primer nivel

Un **nodo hoja** del árbol es marcado como deseable cuando el ítem que representa es accesible actualmente para el usuario, en este caso su visita supone directa o indirectamente un acercamiento del usuario a la satisfacción de sus intereses.

Visualmente representamos que un ítem i_j es deseable marcando con el símbolo √ el nodo hoja etiquetado con dicho ítem en el $Tree_D$. El procedimiento para generar completamente el árbol es el que se especifica a continuación.

Paso 1. Para cada nodo hoja n_j del $Tree_D$ que aún no ha sido marcado como deseable:

Paso 1.1 Se sustituye el nodo n_i por el árbol de accesibilidad $Tree_K(i_i)$ asociado al ítem i_i que representa.

Paso 1.2 Se evalúa el $Tree_K(i_i)$ con el estado de conocimiento que actualmente posee el usuario, esto es $estado_K$'.

Paso 1.2.1 Para cada nodo hoja n_k del $Tree_K(i_i)$ se evalúa el requisito de conocimiento, $K(i_k) \geq val$, asociado a su rama. Y en caso de tratarse de un requisito satisfecho se eliminan el nodo y la rama.

Paso 1.2.2 Para eliminar una rama *and* es necesario que se eliminen todas las subramas que la componen. Lo cual quiere decir, que se ha satisfecho el antecedente de accesibilidad de una regla $Rk(i_j)$.

Paso 1.2.3 Cuando se elimina una subrama *or*, se eliminan también todas las subramas unidas a ésta, eliminándose pues la rama *or* completa. Lo cual significa que el ítem i_j es accesible, ya que lo es, al menos por una $Rk(i_j)$.

Paso 1.3 Si en el paso 1.2 se eliminan todas las ramas del $Tree_K(i_i)$, quedando únicamente el nodo raíz n_i, significa que el ítem i_i es actualmente accesible. Por lo tanto, el nodo n_j se marca como deseable con el símbolo √.

Paso 2. Si todos los nodos hoja del $Tree_D$ están marcados como deseables Terminar, en otro caso volver al Paso 1.

Terminar.

Algoritmo 1. Construcción del $Tree_D$

Siguiendo con nuestro ejemplo, después de obtener el primer nivel del $Tree_D$ mostrado en la figura 3.1, el siguiente paso es sustituir el nodo hoja etiquetado con I3 por su árbol de accesibilidad correspondiente, esto es $Tree_K(I3)$.

Observe, en la figura 3.2, que tras evaluar todos los nodos hoja del $Tree_K(I3)$, únicamente la rama asociada al ítem I4 se elimina porque el conocimiento del usuario sobre I4 es "total", satisfaciendo el requisito impuesto sobre éste, K(I4) ≥"high". Puesto que el subárbol que cuelga de I3 no ha sido eliminado completamente, el nodo I3 no puede marcarse como deseable, quedando el $Tree_D$ en el estado que se muestra.

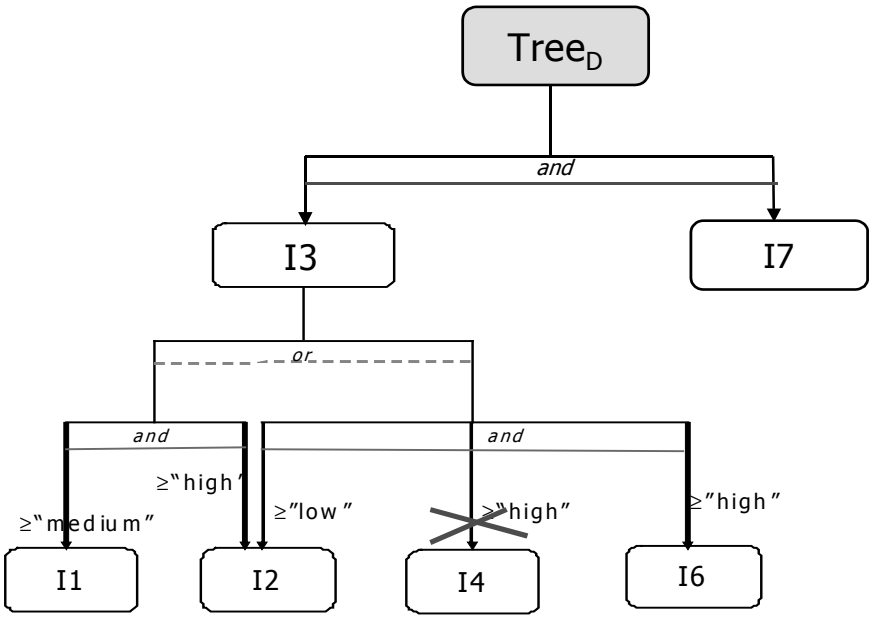


Figura 3.2: 2º paso $Tree_D$ – $Tree_K(I3)$

La figura 3.3 muestra el aspecto que presenta el $Tree_D$ después de sustituir el nodo de primer nivel I7 por el árbol de accesibilidad correspondiente. En este caso, el subárbol $Tree_K(I7)$ tiene una única rama cuyo requisito de conocimiento asociado, K(I4) ≥ "low", es actualmente satisfecho por el usuario.

Por este motivo, tanto el nodo I4 como la rama que llega hasta él se elimina, y el nodo I7 se marca como deseable al quedar probada su accesibilidad. En este caso, el ítem I7 es deseable ya que al visitarlo el usuario está accediendo a un ítem que el mismo ha calificado como interesante.

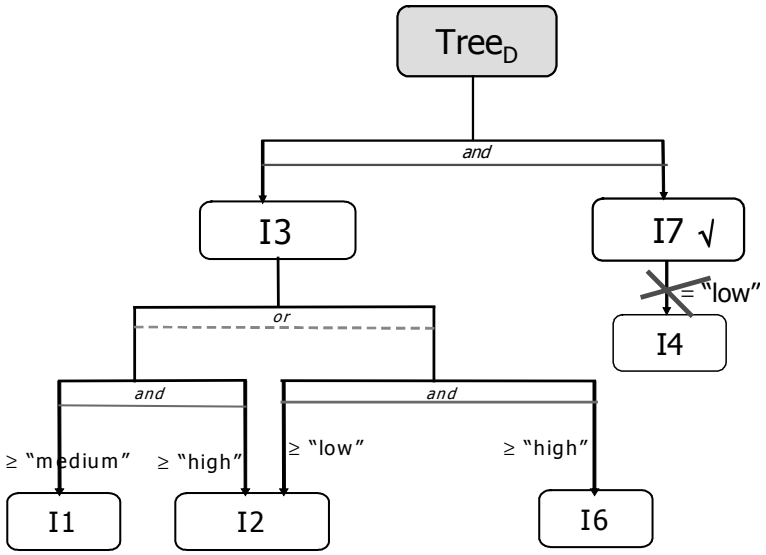


Figura 3.3: 3er paso Tree$_D$ – Tree$_K$(I7)

En la figura 3.4 se muestra el estado final del Tree$_D$ después de realizar una nueva iteración del procedimiento explicado, esta vez para los nodos hoja I1, I2 e I6. Como en el ejemplo no existen restricciones de accesibilidad asociadas a ninguno de ellos (figura 2), los tres ítems son inmediatamente marcados como deseables. Estos ítems no son interesantes, pero su visita es deseable porque contribuye directamente a que el ítem I3, que sí lo es, se vuelva accesible.

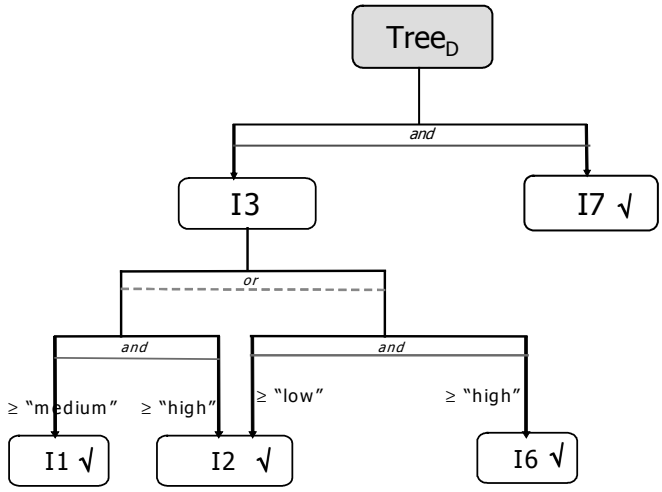


Figura 3.4: Tree$_D$ final

Finalmente en el Tree$_D$ sólo se incluyen los ítems requeridos (y no satisfechos) para que todos los ítems interesantes se conviertan en accesibles y puedan ser visitados por el usuario, tal y como es su deseo. Quedando el conjunto de ítems deseables como Dbls = {I1, I2, I6, I7}.

2.3 Anotación de ítems deseables

Una vez generado el Tree$_D$ y obtenido el conjunto de ítems deseables, el sistema debe informar del resultado al usuario. Puesto que los ítems incluidos en el conjunto pueden

cambiar en cada paso de navegación, consideramos que el modo más adecuado para dar a conocer los ítems deseables es resaltar cada uno de éstos directamente sobre la estructura de navegación.

En la figura 4, se muestra el aspecto de la estructura de navegación de la figura 1 tras anotar sobre ésta el conjunto de ítems deseables obtenido en el ejemplo. Como puede observarse, cada ítem del conjunto, Dbls = {I1, I2, I6, I7}, ha sido resaltado con un círculo que rodea su identificador.

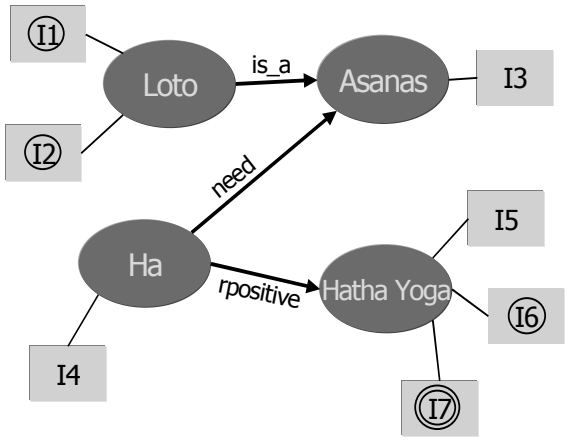


Figura 4: Anotación de ítems deseables

Observe, que en el caso del ítem I7 el círculo es doble. Esta diferencia, nos sirve para indicar al usuario que ese ítem deseable es, en sí mismo, un ítem interesante.

La anotación realizada, independientemente de la forma en que se lleve a cabo, permite orientar al usuario sobre los ítems que en este preciso momento puede visitar para acercarse a sus intereses. Es decir, aunque debido a las restricciones impuestas en las reglas de conocimiento, el usuario no pueda visitar todos los ítems que le interesan, al menos conoce qué ítems puede visitar ahora (ítems deseables), para poder visitar más adelante uno o varios de esos ítems anhelados.

También puede ser útil anotar los ítems deseables para cada ítem interesante. Es decir, marcar de forma diferente o separada los ítems deseables según el ítem interesante por el que han sido marcados de tal forma. De este modo, el usuario puede concentrarse en un determinado ítem interesante, visitando los ítems deseables que permiten conseguir que éste se vuelva accesible.

Conocer para qué ítem interesante i_j es beneficiosa la visita de un ítem deseable i_k es inmediato y automático: simplemente hay que buscar en el $Tree_{D,}$ el nodo de primer nivel n_j del que cuelga el subárbol donde se encuentra el nodo deseable n_k. Por ejemplo, en la figura 3.4, esto permite identificar I1, I2 e I6 como deseables para I3.

Obviamente, para llevar a cabo esta forma de anotación existen alternativas muy diversas. Una de ellas consiste en escribir en la estructura de navegación el nombre de cada ítem interesante en un color diferente. De modo que el círculo que rodea a un ítem deseable presente el mismo color que el ítem interesante del que procede. Esta solución, aunque muy clara visualmente, no es sencilla cuando un ítem deseable es beneficioso

para dos o más ítems interesantes, ya que en este caso el círculo del ítem debe presentar varios colores.

Otras posibilidades son:

- El uso de flechas, preferiblemente discontinuas, que sobrepuestas sobre la estructura de navegación, apunten desde cada ítem deseable a cada uno de los ítems interesantes a los que beneficia.

- Que al realizar un determinado evento, por ejemplo doble *clic*, sobre un ítem interesante se resalten los círculos que rodean a los ítems deseables para el ítem seleccionado, por ejemplo con un borde más grueso.

- O, simplemente una anotación textual junto a cada ítem deseable con los ítems interesantes que lo motivan.

2.4 Actualización del $Tree_D$

Durante la navegación por conocimiento, siempre que un usuario visita un ítem i_j lo hace en posesión del conocimiento previo necesario para asimilar el contenido de éste, y en consecuencia el sistema ejecuta la regla de actualización asociada, $Ru(i_j)$.

Al ejecutar $Ru(i_j)$ sobre el estado de conocimiento actual del usuario, es muy posible que se incremente el grado de conocimiento que éste posee sobre determinados ítems actualizados en el cuerpo de la regla. El cambio en el estado de conocimiento del usuario, si lo hay, es registrado en su modelo de usuario.

Tras esta adquisición de conocimiento por parte del usuario, el sistema tiene que **reevaluar el $Tree_D$**, ya que algunas restricciones incluidas en éste pueden haberse satisfecho tras el cambio, requiriéndose la poda de determinadas ramas del árbol.

Las modificaciones realizadas en el $Tree_D$ para el nuevo estado de conocimiento, pueden dar lugar a que determinados ítems entren y/o salgan del conjunto de ítems deseables, Dbls. Por lo tanto, hasta que todos los ítems interesantes sean deseables, el sistema debe encargarse de actualizar el $Tree_D$ después de cada visita del usuario. El procedimiento empleado para ello se especifica en el algoritmo 2.

Después de que el usuario visite el ítem i_j:

Paso 1. Comprobar si existen ítems interesantes que aún no son deseables. Esto ocurre cuando Ints' ≠ Dbls, o lo que es lo mismo el $Tree_D$ tiene una profundidad mayor que uno.

En caso afirmativo ir al Paso 2, en otro caso Terminar.

Paso 2. Tomar la regla de actualización asociada al ítem visitado.

$Ru(i_j)$: Si Visitado(i_j) entonces Actualización(i_j), Actualización(i_k),..., Actualización(i_m);

Paso 3. Para cada ítem i_k actualizado en el cuerpo de $Ru(i_j)$:

Paso 3.1 Se eliminan en el $Tree_D$ las ramas que llegan hasta su correspondiente nodo, n_k, con una restricción de conocimiento, ≥ val, satisfecha por el grado de conocimiento, $K'(i_k)$, obtenido tras la actualización, Actualización(i_k).

Paso 3.2 Si se eliminan todas las ramas que llegan a un nodo n_k se elimina también el nodo y todo el subárbol que cuelga de él.

Paso 3.3 De nuevo, si se eliminan todos las subramas de una rama *and* o alguna subrama de una rama *or* se elimina la rama completa.

Paso 4. Si durante el paso 3 algún nodo intermedio se ha convertido en hoja se marca como deseable. Y se actualiza el conjunto de ítems deseables con las nuevas hojas del árbol. Dbls = {nodo-hoja(TreeD)}

Terminar.

Algoritmo 2. Actualización del $Tree_D$

En la figura 5 se muestra el estado del $Tree_D$ de la figura 3.4, después de una visita del usuario al ítem I6, suponiendo para ello la siguiente regla de actualización asociada:

Tabla 4: Regla de actualización para I6

Ru(I6): Si Visitado(I6) entonces Fix-abs(I6, "total"), Fix-abs(I2, "low"), Inc-abs(I7,1);

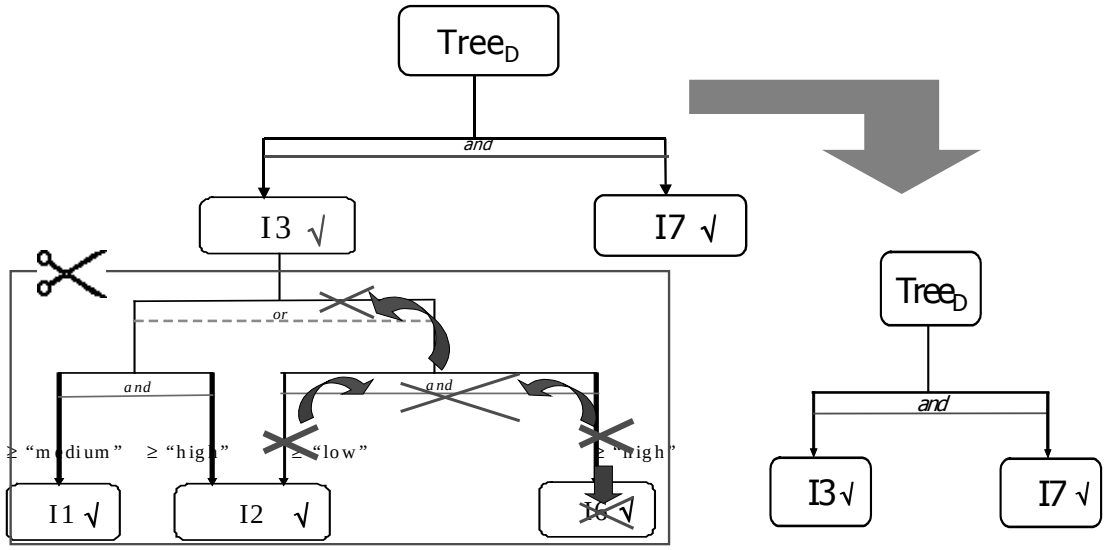


Figura 5: Actualización del $Tree_D$ tras la visita a I6

Después de ejecutar la primera actualización del cuerpo de la regla, Fix-abs(I6, "total"), el grado de conocimiento del usuario sobre el ítem visitado es "total". Este nuevo grado de conocimiento satisface el único requisito asociado en el árbol al nodo I6, ≥ "high", lo que da lugar a la eliminación de la rama y el nodo.

La segunda actualización en Ru(I6) hace que el usuario alcance sobre I2 el conocimiento mínimo requerido, "low", en una de las ramas que llegan a dicho nodo en el árbol. En principio, se elimina sólo la rama, no el nodo, ya que éste cuelga de otra rama etiquetada con un requisito aún no satisfecho, ≥ "high".

Sin embargo, la eliminación de la subrama anterior ocasiona la eliminación de la rama *and* de la que forma parte, y ésta a su vez la eliminación de la rama *or* situada más arriba. De manera que, como puede verse en la figura 5, se produce la eliminación de todo el subárbol que cuelga de I3, lo cual convierte el nodo I3 en hoja y el ítem I3 en deseable.

Tras realizar la poda del $Tree_D$, éste presenta profundidad uno, lo que significa que todos los ítems interesantes son accesibles, y por tanto deseables, Ints' = Dbls = {I3, I7}. Los ítems I1, I2 e I6 han dejado de ser deseables ya que su único objetivo era conseguir que el ítem interesante I3 se convirtiese en accesible, y tras la visita de I6 así ha sido.

El ejemplo anterior, además de mostrar como se actualiza el $Tree_D$, permite dejar de manifiesto que visitar los ítems deseables, es una manera, pero no la única de conseguir que un ítem interesante se vuelva accesible. De hecho, a menudo existen otras posibilidades debido a que las reglas de actualización afectan, además de al ítem visitado, a otros ítems relacionados.

Volviendo al ejemplo anterior, el $Tree_D$ establece que I3 se hace accesible si el usuario conoce con cierto nivel los ítems I2 e I6. Por lo tanto, ambos se marcan como deseables. Está garantizado que visitando una o más veces I2 e I6 se consigue un conocimiento "total" sobre ellos y, en consecuencia, se puede visitar I3.

Sin embargo, la regla de actualización de I6 es tal, que tras una visita a éste se consigue el grado de conocimiento necesario sobre I6 y también sobre I2, con lo que basta para que I3 se vuelva accesible.

Además de la actualización sufrida como consecuencia de la navegación del usuario, el $Tree_D$ debe ser modificado cuando el usuario altera el conjunto de intereses, Ints', utilizado para construirlo. En este caso, es posible distinguir dos situaciones diferentes:

1) El usuario añade un nuevo ítem i_k en el conjunto de intereses Ints'.

2) El usuario elimina un ítem i_k del conjunto de intereses Ints'.

Cuando el usuario elimina la marca de interesante de un ítem i_k **(1)** éste es eliminado del conjunto de ítems interesantes, esto es Ints = Ints – i_k. Si el usuario se encuentra navegando por conocimiento una estructura EC_A^j, donde aparece el ítem i_k éste se elimina también del subconjunto Ints', y como consecuencia el $Tree_D$ es actualizado.

Esta actualización consiste en eliminar del $Tree_D$ la rama de primer nivel que va desde el nodo raíz al nodo etiquetado con el ítem i_k. Además, si el nodo se queda desconectado, es decir, no llega a él ninguna otra rama, éste se elimina del $Tree_D$, con la consecuente poda de todo el subárbol que cuelga de él.

Por el contrario, cuando el usuario marca un nuevo ítem i_k como interesante **(2)**, éste se incluye en el conjunto Ints, Ints = Ints $\cup$ $\{i_k\}$. Si dicho ítem aparece en la estructura que actualmente recorre el usuario en modo de navegación por conocimiento, también se incluye en el conjunto Ints', con la consecuente modificación del $Tree_D$ asociado.

En este caso la modificación consiste en crear una nueva rama, partiendo del nodo raíz y con destino en un nodo etiquetado con i_k. Si dicho nodo ya existe en el $Tree_D$, la adicción de la rama es la única modificación. En otro caso, hay que crear el nodo y aplicar el mismo procedimiento que para el resto (algoritmo 1), es decir, sustituirlo por su $Tree_K$, evaluar las restricciones del $Tree_K$, eliminar las restricciones satisfechas, etc.

El $Tree_D$ cambia en función del estado de conocimiento del usuario, $estado_K$', y de los ítems marcados como interesante por éste, Ints'. Sin embargo, fijados ambos, el $Tree_D$ generado varía dependiendo de las reglas de conocimiento impuestas por el autor. Concretamente un $Tree_D$ puede verse afectado si se modifica el $Tree_K$ asociado a alguno de los ítems que etiquetan sus nodos.

El árbol de accesibilidad, $TreeK(i_j)$, asociado a un ítem i_j cambia cuando el autor añade, elimina o modifica alguna restricción de accesibilidad en una regla de conocimiento definida sobre i_j. También, cuando éste añade o elimina para i_j una regla de conocimiento con antecedente de accesibilidad no vacío (véase el capítulo 8, Reglas de conocimiento).

En definitiva, si cambian los requisitos de accesibilidad en el conjunto de reglas $Rk(i_j)$, y el $Tree_D$ contiene un nodo etiquetado como i_j, éste debe ser reconsiderado. Concretamente, cuando evoluciona $Tree_K(i_j)$ la actualización afecta a todos los usuarios que estuviesen navegando por conocimiento una estructura que incluye el ítem i_j y cuyo estado de conocimiento y conjunto de intereses es tal que dicho ítem se corresponde con un nodo en el $Tree_D$. La modificación del $Tree_D$ es sencilla: reemplazar el subárbol que cuelga del nodo i_j por el nuevo $Tree_K(i_j)$, evaluando sus restricciones de conocimiento sobre $estado_K$' y podándolo en función de los resultados.

La tabla 5 resume los distintos procesos de actualización que pueden tener lugar sobre el $Tree_D$. Por supuesto, todos ellos se realizan de forma automática, independientemente de si los provoca la interacción del usuario (visitar un ítem o modificar su conjunto de intereses), o la acción del autor (modificar el conjunto de reglas de conocimiento).

Tabla 5: Modificación del $Tree_D$

<table>
<tr><th>Acción</th><th>Realizada por</th><th>Causa</th><th>Cambios realizados</th><th>Afecta a</th></tr>
<tr><td>Visitar i_j</td><td rowspan="3">Usuario</td><td rowspan="3">Funcionamiento del sistema</td><td>Se reevalúan las ramas que llegan a un nodo etiquetado con un ítem actualizado en la regla $Ru(i_j)$.</td><td rowspan="3">El $Tree_D$ del usuario implicado</td></tr>
<tr><td rowspan="2">Cambiar Ints</td><td>Añadir ítem interesante i_j: Se añade una nueva rama de primer nivel que tiene como destino el nodo i_j (si no existe se crea).</td></tr>
<tr><td>Eliminar ítem interesante i_j: Se borra la rama de primer nivel que tiene como destino el nodo i_j (si es necesario se elimina también el nodo).</td></tr>
<tr><td>Modificar $Rk(i_j)$</td><td>Autor</td><td>Evolución</td><td>Se reemplaza el subárbol que parte del nodo etiquetado como i_j con el $Tree_K(i_j)$ modificado, eliminando en éste las restricciones ya satisfechas.</td><td>Todos los $Tree_D$ con el nodo i_j</td></tr>
</table>

En la tabla se indica, también, si las modificaciones sobre el $Tree_D$ son consecuencia del propio funcionamiento del sistema (navegación del usuario y actualización explícita de su modelo), o son ejecutadas como parte del propio mecanismo evolutivo (al modificar el conjunto de reglas de conocimiento asociadas a un ítem, el sistema regenera su $Tree_K$ y propaga el cambio a los $Tree_D$ que corresponda).

3. ÍTEMS DESEABLES PARA UNA META DE CONOCIMIENTO

Además de marcar los ítems que le interesa visitar, el usuario puede establecer una meta de conocimiento donde especifique el grado de conocimiento que desea alcanzar sobre cada ítem de un conjunto que le atrae especialmente. Una meta está formada por un conjunto de submetas, donde cada submeta (i_j, et_{SEM}^j) establece el grado de conocimiento mínimo, et_{SEM}^j, que el usuario desea lograr sobre el ítem meta i_j (capítulo 12, Modelo de usuario).

La diferencia entre un ítem interesante (Def 12.4) y un ítem meta (Def 12.7), es que el usuario está interesado en acceder al contenido del primero independientemente de la adquisición de conocimiento que esto le suponga, mientras que respecto al segundo desea alcanzar un grado de conocimiento determinado, sin importarle si éste se alcanza visitando dicho ítem o mediante la visita de otros ítems relacionados.

Puesto que la intención del usuario no es la misma cuando establece los ítems interesantes que cuando define una meta de conocimiento, el procedimiento que se sigue para identificar los ítems deseables a partir de una meta presenta algunas diferencias con el que se ha explicado en la sección anterior.

De nuevo, la meta M' utilizada para obtener los ítems deseables es un subconjunto de la meta M almacenada en el modelo de usuario. Dicho subconjunto está formado por todas las submetas (i_j, et_{SEM}^j) $\in$ M, tales que el ítem sobre el que se definen, i_j, está incluido en la estructura de navegación actual (véase la ecuación 2).

$$(i_j, et_{SEM}^j) \in M' \text{ si } (i_j, et_{SEM}^j) \in M \wedge i_j \in I(EC_P^k) \qquad (2)$$

Partiendo de una meta **M'** = $\{(i_1, et_{SEM}^1),..., (i_n, et_{SEM}^n)\}$ definida sobre los ítems de una estructura EC_p^k, decimos que un ítem i_j perteneciente al dominio de información de esa estructura **es deseable si** satisface dos propiedades:

Propiedad 1. El ítem i_j es un *ítem accesible* dado el estado de conocimiento actual del usuario. Esto es, $estado_K$' satisface las condiciones de acceso en $Rk^r(i_j) \in Rk(i_j)$, y

Propiedad 2. El ítem i_j es un *ítem meta*, es decir está presente en alguna submeta de la meta de conocimiento actual, (i_j, et_{SEM}^j) $\in$ M', o la visita del usuario al ítem i_j *contribuye directa o indirectamente a que un ítem meta se convierta en accesible.*

La principal diferencia con respecto a los ítems interesantes, es que mientras que el conjunto de éstos, Ints', no varía en función de la navegación del usuario. Los ítems meta incluidos en M' pueden disminuir conforme el usuario visita ítems y, como consecuencia, aumenta su conocimiento. Concretamente, una submeta (i_j, et_{SEM}^j) es eliminada de M', M' = M' – $\{(i_j, et_{SEM}^j)\}$, cuando el usuario satisface el requisito impuesto, es decir $K(i_j) \geq$ valor-numérico(et_{SEM}^j).

Por lo tanto, al visitar un ítem deseable, el usuario debe saber que está:

a) visitando un ítem para el que definió una meta de conocimiento que aún no está satisfecha, o

b) visitando un ítem que le acerca al estado de conocimiento requerido para poder visitar uno o varios de los ítems meta.

Al igual que antes, a) y b) se apoyan en el hecho de que las reglas de actualización son definidas de forma que la visita repetida a un ítem garantiza al usuario un conocimiento "total" sobre éste, lo que asegura que se satisface cualquier restricción o meta de conocimiento impuesta para el ítem.

Para poner de manifiesto las diferencias que existen en el proceso de generación del $Tree_D$ para una meta de conocimiento respecto a este mismo proceso en el caso de un conjunto de ítems interesantes, nos serviremos del ejemplo descrito en la sección anterior (figuras 1 y 2). Salvo, claro está, que en esta ocasión en lugar del conjunto de ítems interesantes Ints' = {I3, I7} suponemos una meta de conocimiento M' compuesta por dos submetas asociadas respectivamente a esos dos ítems. Concretamente la meta especificada exige un conocimiento mínimo "total" sobre I3 y "high" sobre I7, M' = {(I3, "total"), (I7, "high")}.

La **construcción del árbol** de ítems deseables comienza nuevamente con la generación de un **nodo raíz** etiquetado como *Tree*$_D$. El **primer nivel** del árbol está formado por tantos nodos como submetas constituyen la meta actual M'. Todos los nodos de primer nivel se unen mediante el operador *and*, indicando que para satisfacer la meta global es necesario satisfacer todas y cada una de la submetas que la componen.

La diferencia es que, en este caso, las ramas del primer nivel son etiquetadas con la restricción de conocimiento impuesta en la submeta correspondiente. Así, la rama que llega al nodo etiquetado como i_j tiene asociada la restricción $\geq et_{SEM}^j$, reflejando la existencia en M' de una submeta (i_j, et_{SEM}^j).

Antes de generar los niveles inferiores del árbol, el sistema evalúa las restricciones de conocimiento que etiquetan el primer nivel del $Tree_D$. Si la restricción asociada a una rama, $\geq et_{SEM}^j$, se satisface en el estado de conocimiento actual del usuario, $K(i_j) \geq$ valor-numérico(et_{SEM}^j), se elimina dicha rama y el nodo n_j al que llega, eliminando también de M' la submeta correspondiente, $M' = M' - \{(i_j, et_{SEM}^j)\}$. En la figura 6 se muestra el primer nivel del $Tree_D$ generado para la meta del ejemplo.

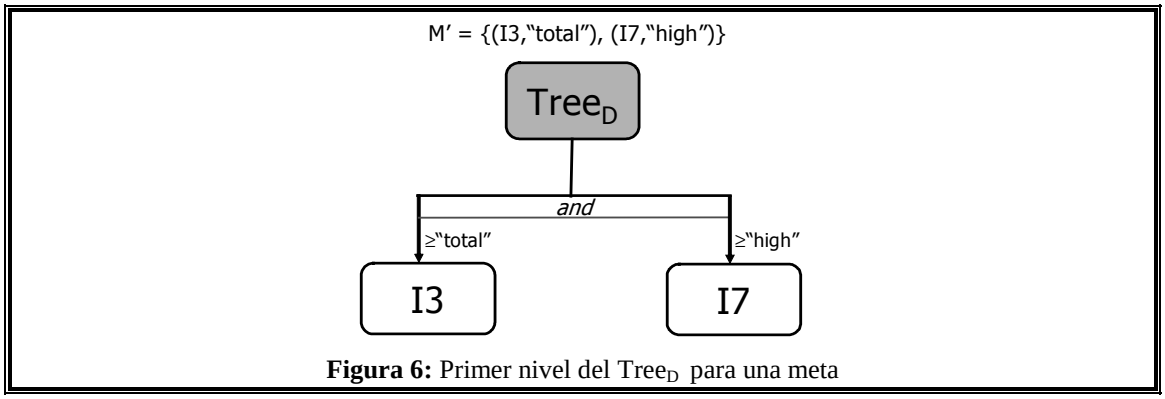


Figura 6: Primer nivel del $Tree_D$ para una meta

A partir del primer nivel del árbol, el procedimiento para construir el resto del $Tree_D$ es idéntico al explicado para el conjunto de ítems interesantes (algoritmo 1). Es decir, se sustituye cada nodo hoja del $Tree_D$ por su árbol de accesibilidad $Tree_K$, eliminando de

este subárbol las restricciones de conocimiento satisfechas actualmente por el usuario, de modo que si se elimina el subárbol completo, el nodo se marca como deseable.

Una vez obtenido el conjunto de ítems deseables, el procedimiento de anotación coincide, también, con el explicado para el conjunto de intereses. La única diferencia es el símbolo que se utiliza para resaltar los ítems del conjunto, en este caso un cuadrado que bordea el identificador del ítem deseable (doblemente si se trata de un ítem meta). Gracias a esta diferencia, el usuario puede distinguir los ítems deseables para la meta de los deseables para el conjunto de intereses, en el caso de que ambos convivan.

Después de aplicar el procedimiento explicado en el algoritmo 1, el árbol generado para la meta de conocimiento M' es el que se muestra en la figura 7. Como puede observar, este árbol difiere únicamente del mostrado en la figura 3.4 en la existencia de restricciones de conocimiento en las ramas del primer nivel.

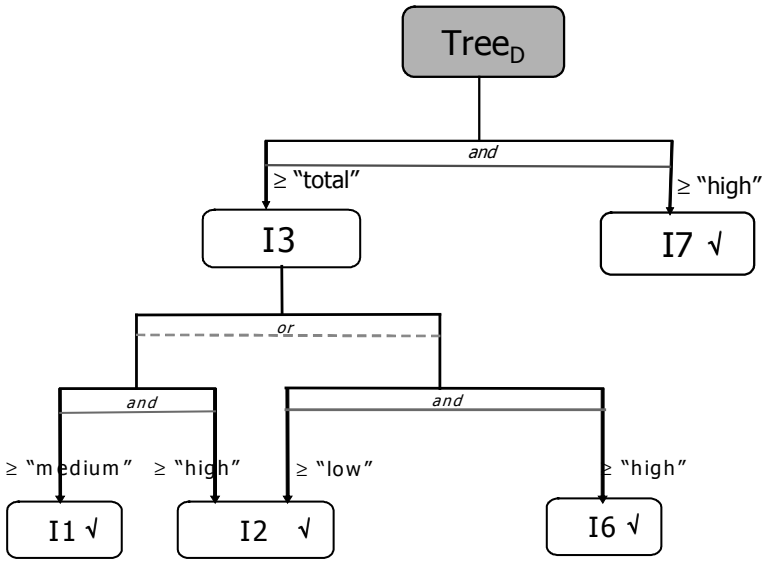


Figura 7: $Tree_D$ para M' = {(I3, "total"), (I7, "high")}

Puesto que los nodos hoja del $Tree_D$ son los mismos, también lo es el conjunto de ítems deseables, I1, I2, I6 e I7, cuya anotación se muestra en la figura 8.

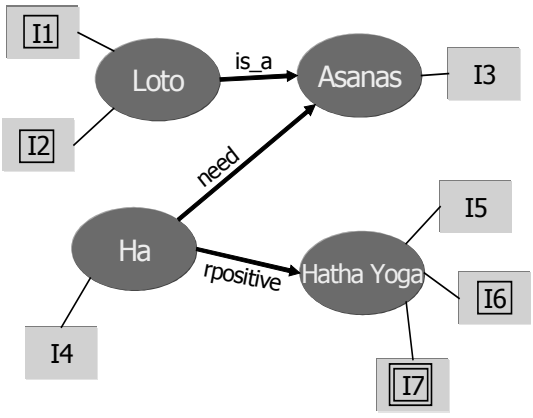


Figura 8: Anotación de ítems deseables para M'

De nuevo, con un simple vistazo el usuario puede identificar que el ítem I7 además de ser un ítem deseable es un ítem meta, siendo consciente de que después de una o varias visitas a éste se asegura la consecución del grado de conocimiento deseado, cualquiera que sea.

Por su parte, el proceso de **actualización del $Tree_D$** que tiene lugar tras cada visita del usuario, difiere del expuesto para los ítems interesantes (sección 2.4), en que ahora también se evalúan las restricciones de conocimiento asociadas a las ramas del primer nivel del árbol.

Mientras que en el caso de un conjunto de ítems interesantes, el $Tree_D$ mantiene indefinidamente como deseable un ítem interesante una vez que este se hace accesible (salvo modificaciones en los intereses del usuario o en las reglas de conocimiento). En el caso de una meta de conocimiento, un ítem meta sólo es deseable desde el momento en que se vuelve accesible hasta el momento en que el usuario alcanza o supera el conocimiento mínimo exigido sobre él.

Dicho de otro modo, el $Tree_D$ generado para un conjunto de ítems interesantes tiene siempre como mínimo un primer nivel que cuelga del nodo raíz con todos los ítems incluidos en el conjunto Ints'. Por el contrario, el $Tree_D$ asociado a una meta de conocimiento queda vacío cuando el usuario ha satisfecho todas las submetas contenidas en la meta especificada, M'.

Las diferencias explicadas, hacen necesario "retocar" el procedimiento de actualización especificado en el algoritmo 2. Ahora, éste se ejecuta aunque todos los ítems meta sean deseables, siendo la condición de parada que la meta esté vacía (paso 1). Además se incluye un paso más (paso 5), para comprobar si durante el paso 3 se ha eliminado alguna rama de primer nivel, lo cual significa que una submeta ha sido satisfecha y debe ser eliminada de M'.

Después de que el usuario visite el ítem i_j:

Paso 1. Comprobar que la meta del usuario no está vacía. Esto es, M' ≠ ∅.

En caso afirmativo ir al Paso 2, en otro caso Dbls = ∅ y Terminar.

Paso 2. Tomar la regla de actualización asociada al ítem visitado.

$Ru(i_j)$: Si Visitado(i_j) entonces Actualización(i_j), Actualización(i_k), ..., Actualización(i_m);

Paso 3. Para cada ítem i_k actualizado en el cuerpo de $Ru(i_j)$:

Paso 3.1 Se eliminan en el $Tree_D$ las ramas que llegan hasta el nodo n_k con una restricción de conocimiento, ≥ val, satisfecha por el nuevo grado de conocimiento, $K(i_k)'$, obtenido tras la actualización, Actualización(i_k).

Paso 3.2 Si se eliminan todas las ramas que llegan a un nodo n_k se elimina también el nodo y todo el subárbol que cuelga de él.

Paso 3.3. Si se eliminan todos las subramas de una rama *and* o alguna subrama de una rama *or* se elimina la rama completa.

Paso 4. Se actualiza el conjunto de ítems deseables con las nuevas hojas del árbol. Dbls = {nodo-hoja($Tree_D$)}

Paso 5. Si durante el paso 3 se ha eliminado una rama de primer nivel asociada a un nodo n_j, se elimina la submeta (i_j, et_{SEM}^j) del conjunto M', esto es M' = M' – $\{(i_j, et_{SEM}^j)\}$.

Terminar.

Algoritmo 3. Actualización del $Tree_D$ para M'

Si suponemos de nuevo que el usuario visita el ítem I6 y se aplica su regla de actualización (tabla 4), las modificaciones sobre el $Tree_D$ de la figura 7 son las que pueden apreciarse en la figura 9.

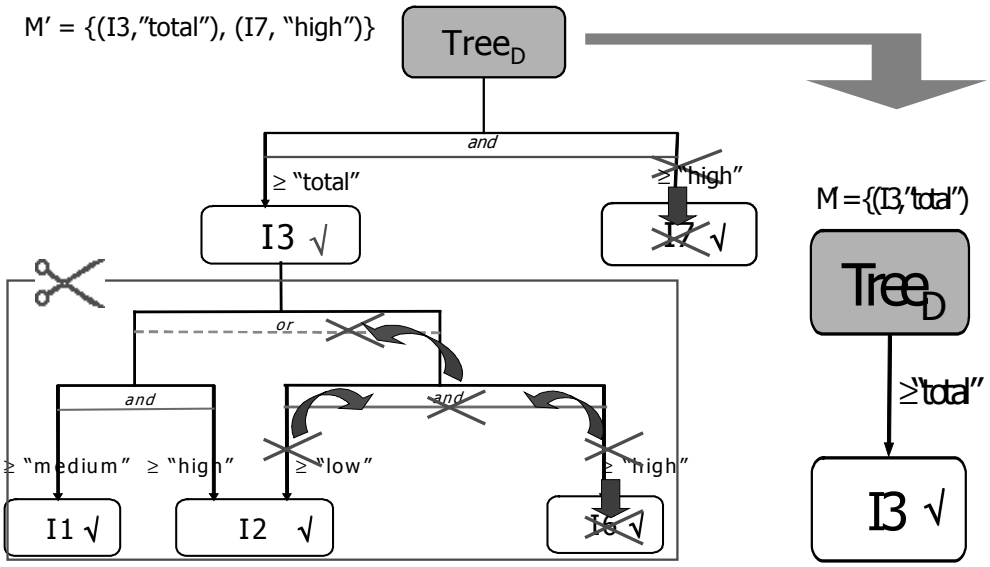


Figura 9: Actualización del $Tree_D$ de una meta tras la visita a I6

En este caso, al igual que ocurría en el ejemplo de la figura 5, el ítem I3 se vuelve accesible al satisfacerse todas las restricciones de accesibilidad de una de sus reglas de conocimiento, Rk^2(I3), representada en el subárbol derecho que cuelga del nodo I3 (rama *or*).

Sin embargo, en este caso además se satisface la restricción de conocimiento asociada a la rama de primer nivel que llega al nodo I7. Esto se debe a que después de visitar I6, el grado de conocimiento del usuario sobre I7 ("medium" antes de la visita) se incrementa en uno, dando como resultado un conocimiento "high" que satisface el requisito impuesto para I7 en la correspondiente submeta. Como consecuencia, el nodo I7 es eliminado del árbol y la submeta (I7, "high") de M'.

De manera idéntica a como acontece en el caso de un conjunto de ítems interesantes, el $Tree_D$ generado para una meta de conocimiento puede verse modificado tras la propagación de un cambio realizado por el autor en el conjunto de reglas de conocimiento definidas para alguno de los ítems que etiquetan sus nodos.

Asimismo, cuando el usuario añade o elimina una submeta en M', el $Tree_D$ generado para ésta, se actualiza de la misma manera (salvo la restricción de conocimiento asociada) que el $Tree_D$ construido para un conjunto de ítems interesantes cuando el usuario añade o elimina un ítem del conjunto.

4. ÍTEMS DESEABLES PARA CONCEPTOS INTERESANTES Y META

En las secciones 2 y 3 se ha explicado el procedimiento que tiene lugar para identificar los ítems deseables a partir de un conjunto de ítems interesantes o ítems meta. En ambos casos, el objetivo que se consigue es el mismo: dar a conocer al usuario un conjunto de ítems que al ser visitados, una o más veces, garantice la satisfacción de su meta o la consecución de sus intereses, según corresponda.

Para ello, en cada momento, el conjunto de ítems deseables contiene ítems accesibles de acuerdo a las reglas de conocimiento, que son requeridos directa o indirectamente para satisfacer alguno de sus deseos (ítem interesante o ítem meta) aún no satisfecho. La cuestión que nos planteamos ahora es, *¿qué pasa cuando entre los deseos del usuario se encuentra acceder a un concepto (concepto interesante) o alcanzar un determinado grado de conocimiento sobre éste (concepto meta)?*

La solución es casi inmediata cuando se trata de un **concepto** marcado como **interesante**, esto es $\mathbf{c_i \in Ints}$. Obviamente, cuando el usuario establece que está interesado en un concepto quiere decir que está interesado en cualquier ítem asociado a éste. Por lo tanto, si el usuario navega una estructura EC_P^k donde el concepto c_i está incluido, el sistema añade al conjunto de ítems interesantes asociado, Ints', todos los ítems ligados funcionalmente al concepto c_i en dicha estructura (véase la ecuación 3).

$$i_j \in \text{Ints'} \text{ si } c_i \in \text{Ints} \wedge c_i \in C(EC_P^k) \wedge <c_i, r_f, i_j> \in Af(EC_P^k) \qquad (3)$$

Por supuesto, si un ítem i_j asociado al concepto interesante c_i ya está incluido en el conjunto Ints' no se vuelve a incluir. Esto es, a i_j le corresponde un único nodo en el $Tree_D$, auque éste sea interesante por si mismo y/o por estar asociado a uno o varios conceptos interesantes.

Supongamos el conjunto de ítems interesantes Ints'={I3, I7} definido sobre la estructura de la figura 1. Si el usuario marca el concepto *Hatha-Yoga* como interesante, automáticamente el conjunto Ints' es ampliado incluyendo los ítems I5 e I6 asociados a éste y no incluidos previamente en Ints' (como es el caso de I7). El resultado es Ints' = {I3, I7, I6, I5}.

Una vez *transformado cada concepto interesante en un conjunto de ítems interesantes*, el procedimiento que se sigue para generar, actualizar y anotar los ítems deseables es el mismo que se ha explicado en la sección 2.

Sin embargo, cuando el usuario establece una **submeta de conocimiento (c_m, et_{SEM}^m) sobre un concepto c_m,** la solución para obtener los ítems deseables no es tan directa como en el caso anterior. Empezando por que, para que un concepto meta c_m sea considerado en una estructura EC_P^k no basta con que c_m esté incluido en dicha estructura, sino que es necesario que el grado de conocimiento impuesto sobre él, et_{SEM}^m, pueda ser obtenido por el usuario durante la navegación de la misma.

Para determinar cuál es el grado de *conocimiento máximo que el usuario puede alcanzar sobre un concepto c_m visitando la EC_P^k*, el sistema necesita conocer el estado de conocimiento actual del usuario, $estado_K$', los ítems asociados a dicho concepto en la EC_P^k y la regla de peso definida para el concepto, $Rw(c_m)$.

Este conocimiento máximo, al que notamos $\mathbf{K_{max}^{k}(c_m)}$, se obtiene aplicando la regla de peso $Rw(c_m)$, considerando para ello, el conocimiento que actualmente posee el usuario sobre los ítems no incluidos en la EC_P^k y conocimiento "total" sobre cada uno de los ítems asociados a c_m en la EC_P^k.

Esta última consideración es posible gracias a que el conjunto de reglas de conocimiento y actualización definido sobre una EC_P^k en cualquier estructura de aprendizaje, siempre garantiza que el usuario puede obtener conocimiento "total" sobre cada uno de los ítems incluidos en su dominio de información (capítulo 10, Consistencia $Rk \cup Ru$).

Respecto a la primera consideración, es evidente que si un ítem no está incluido en EC_P^k es imposible que el usuario aumente su conocimiento sobre éste navegando dicha estructura. Aunque las reglas de actualización permiten incrementar el conocimiento de determinados ítems sin necesidad de visitarlos, no conciben la actualización de un ítem que no aparece en la estructura sobre la que se definen (capítulo 9).

En definitiva, para que un concepto meta c_m sea considerado en la construcción del $Tree_D$ asociado a una estructura EC_P^k, es necesario que el concepto forme parte de su dominio conceptual y que el conocimiento máximo que el usuario puede lograr sobre él supere o iguale el grado de conocimiento requerido en su submeta (véase la ecuación 4).

$$(c_m, et_{SEM}^{m}) \in M' \text{ si } (c_m, et_{SEM}^{m}) \in M \wedge c_m \in C(EC_P^k) \wedge K_{max}^{k}(c_m) \geq \text{valor-numérico}(et_{SEM}^{m}) \quad (4)$$

Partiendo del ejemplo de la figura 1 supongamos que el usuario, dada la meta M' = {(I3, "total"}, (I7, "high")}, añade la submeta (Hatha-Yoga, "high").

Supongamos, también, que la regla de peso asociada al concepto *Hatha-Yoga* es la siguiente:

Rw(Hatha-Yoga) := ¼ K(I5) + ¼ K(I6) + ¼ K(I7) + ¼ K(I8)

Puesto que I5, I6 e I7 aparecen asociados al concepto *Hatha-Yoga* en la estructura del ejemplo, consideramos K(I6) = K(I7) = K(I8) = 4 ("total"). El ítem I8 no aparece en la estructura actual, por lo tanto, es necesario consultar el grado de conocimiento que el usuario posee actualmente en su modelo de usuario. Suponiendo que éste es nulo, o sea K(I8) = 0 ("null"), el resultado que obtenemos tras ejecutar la regla de peso es:

K_{max}(Hatha-Yoga) := ¾ × 4 = 3 ("high")

El conocimiento máximo que este usuario puede obtener sobre el concepto *Hatha-Yoga* recorriendo la EC_P actual satisface la meta impuesta, por lo tanto, el conjunto M' queda como sigue: M'= {(I3, "total"}, (I7, "high"), (Hatha-Yoga, "high")}.

Aunque la meta se define a nivel de concepto, y como tal se incluye en M', el $Tree_D$ se sigue construyendo a nivel de ítem, porque al final lo que permite lograr la meta impuesta sobre un concepto es la adquisición de conocimiento sobre los ítems de información asociados a éste.

Por ello, el concepto meta c_m se representa en el $Tree_D$, no como un nodo, sino como un subárbol de profundidad uno, cuya raíz se etiqueta con el nombre del concepto y cada nodo hijo representa uno de los ítems asociados al concepto en la estructura actual.

En dicho subárbol, la rama que une el concepto raíz con cada nodo hijo, se etiqueta con la restricción $\geq$ "*total*", lo que significa que el nodo se elimina en cuanto que el usuario consiga el conocimiento máximo sobre el ítem que representa. Esto se hace, porque aunque la meta del concepto no se haya alcanzado aún, la visita a ese ítem ya no permite aumentar más el conocimiento del concepto. Observe el subárbol para el concepto *Hatha-Yoga* en la figura 10.

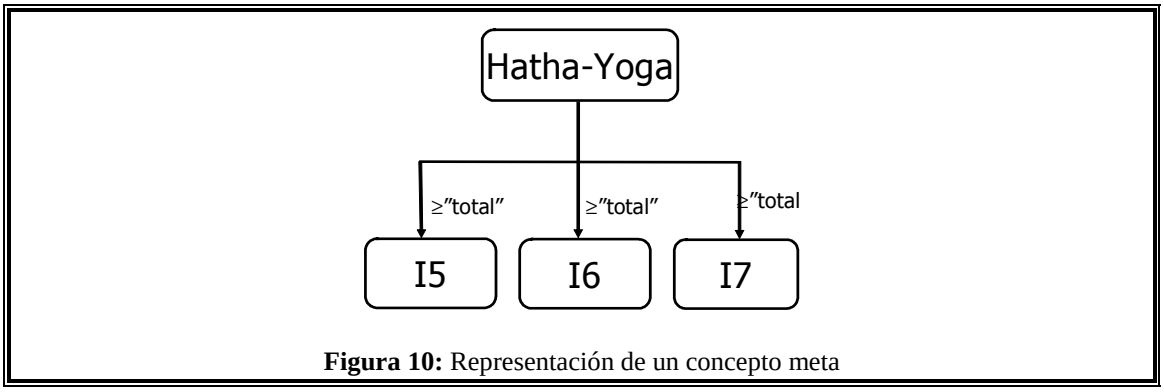


Figura 10: Representación de un concepto meta

El subárbol que representa un concepto meta c_m es incluido en el $Tree_D$ de la misma forma que el nodo que representa un ítem meta (véase la figura 11). Esto es, en el primer nivel del árbol y con la restricción de conocimiento exigida, $\geq et_{SEM}^{m}$, etiquetando la rama que lo une al nodo raíz. A partir de ahí la generación del árbol se realiza de acuerdo a lo explicado en el algoritmo 1.

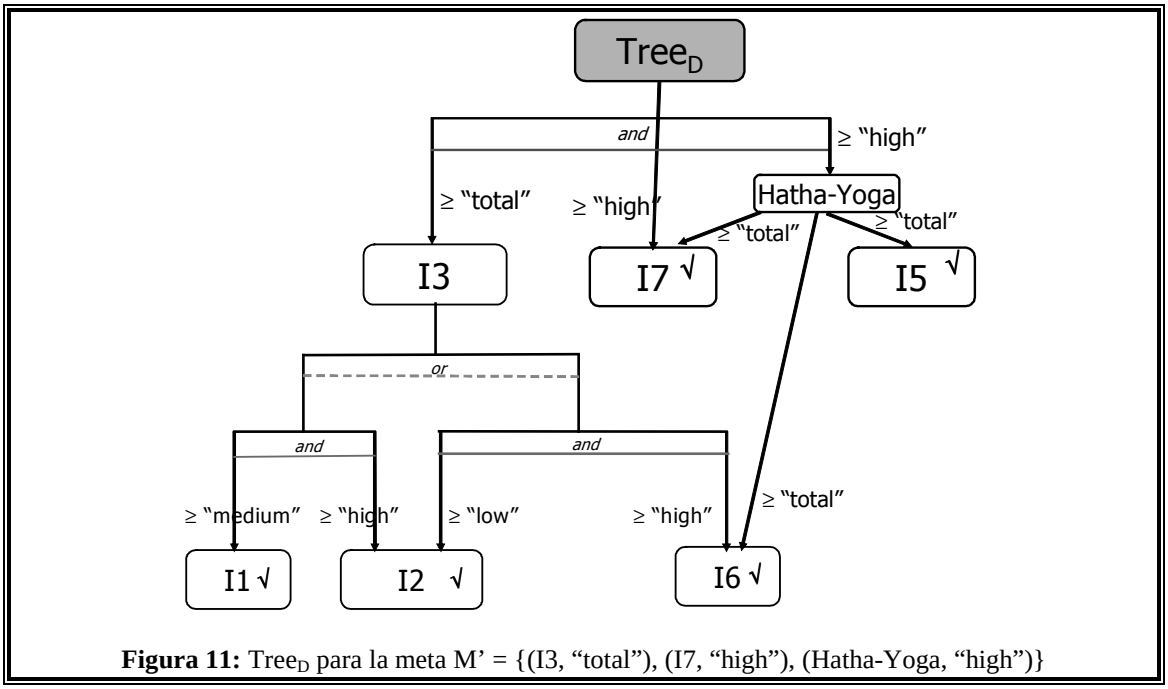


Figura 11: $Tree_D$ para la meta M' = {(I3, "total"), (I7, "high"), (Hatha-Yoga, "high")}

La actualización tras cada visita del usuario se realiza de acuerdo al algoritmo 3, pero añadiendo un nuevo paso que se encarga de aplicar la regla de peso $Rw(c_m)$ de un concepto meta c_m cada vez que el ítem visitado actualiza a alguno de los ítems asociados

a éste. De forma que si el valor obtenido supera la restricción asociada, $K(c_m) \geq et_{SEM}{}^{m}$, se elimina la rama, el nodo meta y todo el subárbol que cuelga de él.

De este modo, los nodos hijos de un concepto meta, se eliminan automáticamente cuando la meta definida sobre el concepto se satisface (salvo que cuelguen de otra parte), aunque la restricción de conocimiento impuesta en su rama, ≥ "total", no se haya satisfecho aún.

CAPÍTULO 20

Rutas Guiadas

Resumen

Una ruta guiada es una secuencia de visitas que conduce al usuario desde su estado de conocimiento actual hasta uno que satisface todas sus metas. Los caminos posibles hasta la meta se trazan en el conjunto de reglas de conocimiento y actualización, y la construcción de la ruta es un problema de búsqueda. En este capítulo se especifica formalmente dicho problema, y se proponen y evalúan distintas soluciones. Para ello, se concibe el espacio de estados como un árbol de búsqueda, resaltando sobre éste una serie de características que resultan fundamentales para garantizar el adecuado funcionamiento de las estrategias, heurísticas y algoritmos planteados.

Tabla de contenidos

Rutas Guiadas

1. INTRODUCCIÓN

En esta sección se plantea la necesidad de una técnica de consejo global, justificando el carácter esencialmente local de la anotación de ítems deseables. En consecuencia se propone la generación de rutas guiadas y se esbozan algunas de sus características y objetivos.

1.1 Necesidad de una técnica de consejo global

En el capítulo 19 (**Ítems deseables**) se describe una técnica que permite adaptar la estructura de navegación a la meta de conocimiento especificada por el usuario. Se trata de una técnica de **consejo local** [Brusilovsky, 96], ya que en cada paso de navegación, el sistema aconseja al usuario qué hacer en el paso siguiente, pero no adelanta nada acerca de los pasos posteriores.

Concretamente, el sistema propone al usuario un conjunto de ítems entre los cuales debería encontrarse el ítem que visite a continuación. Dichos ítems son denominados deseables y el usuario los percibe a través de la anotación realizada sobre la propia estructura de navegación.

Cada vez que el usuario selecciona uno de los ítems marcados como deseables, consigue como resultado una aproximación a su meta de conocimiento. No obstante, el usuario es libre de visitar el ítem que quiera, deseable o no, siempre que éste sea accesible. Es decir, la anotación de ítems deseables, no es vinculante, se trata de una sugerencia que el usuario puede seguir o no, y en caso de hacerlo tiene libertad para elegir cualquiera entre los ítems que se le aconsejan.

De este modo, el sistema no conoce el ítem que el usuario va a visitar en un determinado paso de navegación. Y, sin embargo, como vimos en el capítulo 19, el conjunto de ítems deseables para el siguiente paso depende de esa elección. Por lo tanto, el sistema no puede pronosticar los ítems deseables más allá del paso de navegación actual.

Pese al carácter local de esta técnica, el usuario podría aprovechar la anotación de ítems deseables realizada en un determinado paso de navegación como un **consejo global** para lograr su meta de conocimiento. Esto se debe a que, tal y como ha sido obtenido el conjunto de ítems deseables, se garantiza que visitando cada uno de ellos, una o más veces, en el orden que sea, se satisfacen todos los requisitos de conocimiento impuestos en la meta.

Aunque de esta forma se garantiza que el usuario alcanza su meta en un número finito de pasos, no se puede asegurar que el camino que sigue antes de lograrla sea óptimo, ni siquiera aceptablemente bueno. Las características que motivan esta situación son los siguientes:

a) En la construcción del $Tree_D$ la única forma de obtener conocimiento sobre un ítem que se considera es la visita directa a éste, aunque también pueda obtenerse visitando otros ítems.

b) Cuando un ítem no es accesible se tienen en cuenta todas las reglas de conocimiento asociadas a éste, sin embargo basta con que el ítem se vuelva accesible por una sola de ellas.

La **característica a)** permite que el calculo de los ítems deseables sea inicialmente más *rápido* y sencillo, al no considerarse las reglas de actualización en la construcción del $Tree_D$, sino sólo para su actualización tras cada visita. Sin embargo, esto puede dar lugar a la existencia de ítems *redundantes*, es decir, un ítem deseable cuya visita produce un acercamiento a la meta que ya es producido por otro de los ítems del conjunto.

Por ejemplo, si se requiere que el conocimiento sobre un ítem i_j sea mayor que "high", y el ítem i_j es accesible, éste se incluye como deseable. Esto es redundante si ya existe en el conjunto de deseables un ítem i_k, incluido por otro motivo, que actualiza a "total" el conocimiento sobre i_j en su regla de actualización, $Ru(i_k)$. Por supuesto, cuando el usuario visita i_k, el ítem redundante i_j desaparece.

La **característica b)** concede más *libertad de elección* al usuario que puede hacer accesible un ítem por cualquiera de los caminos posibles. Pero, nuevamente, amplía de forma *redundante* el conjunto de ítems deseables, ya que sólo es necesario que el usuario satisfaga las restricciones de accesibilidad de una de las reglas de conocimiento asociadas al ítem.

Por ejemplo, si para poder acceder a un ítem i_j es necesario conocer más i_1 e i_2 o i_3 e i_4, en el conjunto de ítems deseables se incluyen i_1, i_2, i_3 e i_4, cuando en realidad, a efectos prácticos, se podría haber incluido solamente i_1 e i_2 o i_3 e i_4. De nuevo, cuando el usuario visita i_1 e i_2 los ítems redundantes i_3 e i_4 desaparecen y viceversa.

En resumen, aunque el usuario puede alcanzar su meta visitando todos los ítems anotados como deseables en un determinado paso de navegación, debe saber que eso, generalmente, no es lo óptimo. Es mejor que utilice dicha información con la intención que le ha sido proporcionada, esto es, decidir qué ítem visita a continuación (consejo local), y no como una secuencia de visitas para llegar a la meta (consejo global), ya que, entre otras cosas, los ítems deseables son suministrados sin ningún orden entre ellos.

1.2 Rutas guiadas personalizadas

Una vez considerada la posibilidad de usar la anotación de ítems deseables como técnica de consejo global y descartada su idoneidad, se ha estudiado la forma de dotar al sistema de capacidad para generar una **secuencia ordenada de ítems** que conduzca al usuario, a través de un camino más o menos directo, a la consecución de su meta.

Finalmente, la técnica de consejo global implementada en el Sistema de Aprendizaje consiste en la generación de **rutas guiadas personalizadas** para cada usuario [Medina, 03].

Def 20.1 [Ruta guiada] Una ruta guiada es una secuencia de ítems, esto es, una lista de ítems y un orden estricto para visitarlos. La ruta es generada por el sistema a partir de: el estado de conocimiento actual del usuario, la meta especificada por

éste y sus preferencias. Y siempre se cumple que, después de recorrer la ruta, el estado de conocimiento del usuario satisface completamente la meta.

Gracias a esta técnica, cuando la navegación del usuario está orientada a la consecución de una meta de conocimiento, éste puede solicitar al sistema una ruta adecuada para lograr su objetivo. Y, una vez obtenida, tan sólo preocuparse de visitar sus ítems, uno tras otro, en el orden especificado.

En este caso, el usuario *pierde libertad*, ya que no es él, sino el sistema quien construye la estrategia de navegación completa que debe seguir para llegar a la meta; frente a la anotación de ítems deseables que simplemente le aconseja cuáles de entre los ítems actualmente accesibles es mejor visitar a continuación.

A cambio, cuando el usuario sigue una ruta generada por el sistema *ve incrementada la eficiencia y calidad* de su proceso de navegación. Al referirnos a la **calidad de una ruta** no nos centramos únicamente en el número de pasos que ésta presenta, sino también en las características de los ítems que la forman. Una ruta tiene más calidad en cuanto que su *número de pasos es reducido* y los ítems incluidos *se ajustan a las preferencias* especificadas por el usuario.

Estas **preferencias** se encuentran almacenadas en el modelo de usuario (capítulo 12) y hacen alusión a sus gustos respecto a las *propiedades de los ítems*: idioma, medio, autor, dificultad, rol y fecha, así como a la importancia que éste concede a la longitud de la ruta (atributo *ruta-corta*).

La calidad de una ruta supone, por tanto, un equilibrio entre la longitud de ésta y las características de sus ítems. Por ejemplo, cuando la preferencia del usuario por ruta-corta es baja, no se considera buena una ruta con muy pocos ítems si éstos son desafortunados. Del mismo modo, cuando la preferencia por ruta-corta es alta, no es adecuada una ruta innecesariamente larga, a pesar de que sus ítems se ajusten muy bien a los gustos del usuario.

Durante la generación de la ruta guiada, el sistema puede verse como un **agente basado en metas** [Russell, 96], cuya misión es determinar una *secuencia de acciones* que permita al usuario obtener un *estado deseable*. Las acciones son, por supuesto, visitas a ítems y el estado deseable es cualquiera que satisfaga la meta de conocimiento impuesta.

Afortunadamente en este problema se conoce perfectamente el estado actual y el resultado de las acciones. El *estado de conocimiento actual* se puede consultar en el modelo del usuario (capítulo 12) y el resultado de cada acción se obtiene ejecutando la *regla de actualización* del ítem visitado sobre el estado actual (capítulo 9).

Por tanto, se trata de un **problema de un sólo estado** [Rusell, 96], donde se conoce con exactitud el resultado producido por cada una de las acciones realizadas, pudiendo calcular también con absoluta precisión en qué estado se encuentra el usuario después de una determinada secuencia de acciones.

Las acciones, o sea las visitas a ítems, son las causantes de la transición de un estado de conocimiento a otro. Por lo tanto, una ruta guiada debe proporcionar una secuencia de acciones que permita al usuario obtener un estado que satisfaga su meta de

conocimiento. Para encontrar la solución al problema, es decir la ruta, el sistema debe realizar una *búsqueda a través del conjunto de estados posibles*.

Durante la búsqueda, el sistema encuentra ante sí diversas opciones inmediatas cuyo valor ignora. Para decidir lo que debe hacer primero tiene que evaluar las diferentes secuencias de acciones que le conducen a estados cuyo valor conoce y luego decidirse por la mejor según el criterio de calidad mencionado.

2. ESPECIFICACIÓN FORMAL DEL PROBLEMA

Antes de entrar de lleno en el proceso de generación de la ruta, es conveniente especificar formalmente el problema que se desea resolver. A continuación se especifica cada uno de los elementos que lo componen: estado inicial, operadores, función subsecuente, espacio de estados, meta, prueba de meta, ruta, costo de ruta y solución.

2.1 Estado inicial

El estado inicial es el punto de partida en el proceso de generación de la ruta. En nuestro problema, el estado inicial no es otro que el **estado de conocimiento actual del usuario**. En dicho estado, notado $\mathbf{s_o}$, únicamente se incluye el grado de conocimiento que el usuario posee sobre cada uno de los ítems mostrados en la estructura de navegación actual. Por lo tanto, realmente es un subconjunto del estado de conocimiento completo almacenado en el modelo del usuario, $estado_K$.

Es decir, si el usuario solicita una ruta guiada cuando se encuentra navegando en modo por conocimiento una estructura de aprendizaje $EC_A^j = (EC_M, Rw, MU, EC_P^k, RTnb^i, Ro_i, Ru^j, Rk^j)$, el conocimiento sobre un ítem i_j sólo forma parte del estado inicial, $K(i_j) \in s_o$, si i_j pertenece al dominio de información de la presentación asociada, esto es $i_j \in I(EC_P^k)$ (véase la ecuación 1).

$$s_o \subseteq estado_K \qquad (1)$$

$$s_o = (K(i_1), K(i_2),..., K(i_n)) \text{ donde } i_1,i_2, ...,i_n \in I(EC_p^k)$$

Como puede observar, en la representación del estado inicial, y en general de cualquier estado de conocimiento, se utiliza el valor numérico correspondiente a cada grado de conocimiento en lugar de la etiqueta semántica equivalente. Adviértase, también, que para simplificar la representación del estado se ha usado $K(i_j)$ en lugar de $(i_j, K(i_j))$ como se definió en Def 8.2 (1).

2.2 Operador

El término operador denota la descripción de una acción en función de la cual se alcanza un estado s_{k+1} al emprender dicha acción en un estado particular s_k. En el problema de generación de rutas, el conjunto de acciones posibles se reduce a la **visita de ítems** en la estructura de navegación actual, EC_A^j.

Como consecuencia de la visita a un ítem i_j el sistema ejecuta la regla de actualización de dicho ítem, $Ru(i_j) \in Ru(EC_A^j)$, sobre el estado de conocimiento actual del usuario, s_k,

dando como resultado un nuevo estado, s_{k+1}, donde se habrá incrementado el grado de conocimiento acerca de determinado ítems.

Notaremos **visit(i_j)** al operador que realiza una visita sobre el ítem i_j y **Ru(i_j)[s_k]** al resultado de ejecutar la regla de actualización asociada al ítem i_j sobre el estado de conocimiento representado en s_k (véase la ecuación 2).

$$\text{visit}(i_j): s_k \rightarrow s_{k+1} \quad \text{donde } s_{k+1} = \text{Ru}(i_j)[s_k] \qquad (2)$$

Dada la regla de actualización Ru(i_j) y el estado actual s_k, el nuevo estado s_{k+1} obtenido como consecuencia de la acción visit(i_j) puede diferir del estado s_k en tantos elementos como ítems se actualizan en el cuerpo de la regla Ru(i_j) (véase la ecuación 3).

$$\text{Ru}(i_j): \text{Si Visitado}(i_j) \text{ entonces Actualización}(i_j),..., \text{Actualización}(i_m); \quad (3)$$

$$s_k = (K(i_1),..., K(i_j),..., K(i_m),..., K(i_n))$$

$$s_{k+1} = (K(i_1),..., \text{Actualización}(i_j)[s_k],..., \text{Actualización}(i_m)[s_k],..., K(i_n))$$

Observe en la ecuación 3, que notamos **Actualización(i_j)[s_k]** al grado de conocimiento que se obtiene sobre i_j después de aplicar la actualización establecida en el predicado Actualización(i_j) sobre el grado de conocimiento K(i_j) almacenado en el estado s_k.

2.3 Función subsecuente

La función subsecuente **S**, aplicada sobre un estado particular s_x devuelve el conjunto de estados $\{s_x^1, s_x^2,..., s_x^r\}$ que es posible alcanzar a partir de s_x mediante la ejecución de una sola acción.

En el problema que nos ocupa, el conjunto devuelto por la función S(s_x) tiene tantos estados, esto es *r*, como ítems son accesibles en el estado de conocimiento representado en s_x. Es decir S(s_x) devuelve un estado por cada ítem que podría visitar durante su navegación por conocimiento un usuario con dicho estado de conocimiento.

Dicho de otro modo, si en el estado s_x existen *r* ítems i_1, i_2,..., i_r para los que se satisfacen sus requisitos de accesibilidad, existen también *r* estados en S(s_x). Siendo cada uno de esos estados, s_x^k, el resultado de ejecutar la regla de actualización del ítem accesible i_k sobre el estado actual, $s_x^k = \text{Ru}(i_k)[s_x]$ (véase la ecuación 4).

$$\forall_{k:1..r}\ s_x^k \in S(s_x) \text{ si visit}(i_k): s_x \rightarrow s_x^k \text{ y } i_k \text{ es accesible en } s_x \text{ según } Rk(i_k) \in Rk(EC_A^j) \quad (4)$$

2.4 Espacio de estados

El espacio de estados del problema viene definido por los términos anteriores, ya que es el conjunto de todos los estados que pueden alcanzarse a partir del estado inicial mediante cualquier secuencia de acciones.

En el caso de un estado inicial de desconocimiento absoluto, es decir conocimiento "nulo" sobre todos los ítems, s_o = (0, 0,..., 0), el espacio de estados recoge todos los posibles estados de conocimiento en los que un usuario que se inicia con esa estructura de navegación puede llegar a encontrarse en algún momento.

Gracias a las propiedades exigidas al conjunto de reglas $Rk(EC_A^j) \cup Ru(EC_A^j)$ (capítulo 10), cualquiera que sea el estado inicial, o sea cualquiera que sea el conocimiento del usuario cuando comienza a navegar la estructura EC_A^j, el espacio de estados siempre contiene un estado que representa el conocimiento absoluto, esto es, conocimiento "total" sobre cada uno de los ítems de la estructura, $s_k = (4, 4,..., 4)$.

2.5 Meta

La meta es el estado de conocimiento que el usuario desea alcanzar (Def 12.5). La meta completa de cada usuario, M, se almacena en su modelo. Tal y como se expuso en el capítulo anterior, para obtener la meta **M'** asociada a una determinada estructura de navegación, únicamente se tienen en cuenta las submetas establecidas sobre los ítems incluidos en ésta (véase la ecuación 5).

$$(i_j, et_{SEM}^j) \in M' \text{ si } (i_j, et_{SEM}^j) \in M \wedge i_j \in I(EC_P^k) \qquad (5)$$

Como se observa en la ecuación 5, el usuario establece su meta indicando para cada ítem i_j el grado de conocimiento mínimo, et_{SEM}^j, que desea obtener sobre él. Cuando especifica su meta, el usuario no está obligado a indicar un grado de conocimiento para cada ítem, sino sólo para aquellos que le interese. Esto hace que, normalmente, no exista en M' una submeta (i_j, et_{SEM}^j) para todo ítem i_j incluido en el dominio de información de la estructura actual.

Una meta *M' puede ser satisfecha por más de un estado*, debido a que:

a) Pueden existir ítems cuyo conocimiento es indiferente, esto es, $\exists i_j \in I(EC_P^k)$ tal que $(i_j, et_{SEM}^j) \notin M'$.

b) Cuando el conocimiento mínimo exigido en una submeta, (i_j, et_{SEM}^j), es distinto de "total", ésta puede ser satisfecha por varios grados de conocimiento. Concretamente por todos los grados igual o mayores al exigido, esto es $K(i_j) \geq$ valor-numérico(et_{SEM}^j).

Una meta puede ser satisfecha por varios estados de conocimiento distintos. Concretamente, $M' = \{(i_s, et_{SEM}^s),..., (i_t, et_{SEM}^t)\}$ es satisfecha por cualquier estado de conocimiento s_m que se ajuste a la siguiente plantilla:

$$s_m = (\_,...,\_,K(i_s) ,\_,...,\_, K(i_t), \_,...,\_)$$

$$\text{, donde } K(i_s) \geq \text{valor-numérico}(et_{SEM}^s), ..., K(i_t) \geq \text{valor-numérico}(et_{SEM}^t)$$

, y el símbolo **_** indica que el grado de conocimiento sobre el ítem i_k que ocupa esa posición no importa al usuario y por lo tanto es indiferente el valor que presente:

$$K(i_k) = \_ \equiv K(i_k) \geq 0 \equiv K(i_k) \in [0, 1, 2, 3, 4]$$

Si el usuario no define ninguna submeta sobre un ítem i_k, significa que le parece bien cualquier conocimiento que alcance sobre ese ítem. La ausencia de meta sobre i_k, puede ser entendida como la existencia de la submeta (i_k, "nulo"), donde al exigir que el conocimiento sobre i_k sea mayor o igual que "nulo" se acepta cualquier grado de conocimiento como válido.

2.6 Prueba de meta

La prueba de meta es lo que el sistema *aplica a la descripción de un estado para decidir si se trata de un estado meta* o no. Aunque puede tratarse de un conjunto explícito de estados, en este caso es más sensato utilizar como prueba de meta una **propiedad abstracta**.

Esta propiedad abstracta establece que un estado es meta sólo si presenta un grado de conocimiento sobre cada ítem igual o mayor que el exigido en la submeta asociada a éste. Dado un estado de conocimiento, la prueba de meta sólo tiene en cuenta el grado de conocimiento para el subconjunto de ítems incluido en la meta.

La prueba de meta es realizada por una función que comprueba la propiedad abstracta mencionada y que, por tanto, necesita dos argumentos, un estado de conocimiento, s_k, y una meta M'. El valor de dicha función, **prueba-meta(s_k, M')** es un valor lógico: *true* para indicar que el estado s_k satisface la meta M' y *false* en caso contrario.

De acuerdo a lo anterior, dada una meta M' = $\{(i_s, et_{SEM}^{s}),..., (i_t, et_{SEM}^{t})\}$ y el estado s_k = $(K(i_1), K(i_2), ..., K(i_n))$, la prueba de meta se define como se muestra en la ecuación 6.

$$\text{prueba-meta}(s_k, M') = \text{true si:} \quad (6)$$

$$\forall (i_j, et_{SEM}^{j}) \in M', K(i_j) \in s_k \text{ cumple que: } K(i_j) \geq \text{valor-numérico}(et_{SEM}^{j})$$

2.7 Ruta

Una ruta del espacio de estados es simplemente una **secuencia de acciones** que permite pasar de un estado inicial, s_i, a un estado final, s_f. En nuestro problema, una ruta es cualquier *secuencia de visitas sobre los ítems de una estructura de navegación*. De modo que un usuario que parte de un estado de conocimiento s_i, tras realizar la ruta alcanza un estado de conocimiento s_f.

La longitud de la ruta es el número de acciones, visitas en este caso, que se realizan en la misma. Las acciones de la ruta están ordenadas y se encadenan de forma que el estado resultante de ejecutar una acción constituye el estado sobre el que se va a ejecutar la siguiente. De este modo, el estado final de la ruta, s_f, varía en función de cuál sea el estado inicial, s_i, de la misma. En la ecuación 7 puede verse una la definición de una ruta genérica de longitud *n*.

$$\textbf{ruta}: \text{visit}(i_1), \text{visit}(i_2),..., \text{visit}(i_n) \quad (7)$$

$$\text{visit}(i_1): s_i \rightarrow s_{i+1}\ ;\ \text{visit}(i_2): s_{i+1} \rightarrow s_{i+2}\ ;\ ...;\ \text{visit}(i_n): s_{i+(n-1)} \rightarrow s_f \ \equiv\ \text{ruta}: s_i \rightarrow s_f$$

2.8 Solución

La solución del problema consiste en generar una ruta que permita al usuario, partiendo de su estado de conocimiento actual, s_o, lograr un estado de conocimiento s_m que satisfaga la prueba de meta. Es decir, visitando la información especificada en la estructura de navegación actual, el usuario logra un conocimiento igual o superior al deseado (véase la ecuación 8).

solución: visit(i_1), visit(i_2),..., visit(i_n) (8)

, donde solución: $s_o \rightarrow s_m$ y prueba-meta(s_m, M') = true

2.9 Costo de ruta

2.9.1 Costo de la ruta

La función costo de ruta es una función mediante la que se asigna un costo a una ruta determinada. Este costo es la suma de los costos de cada una de las acciones individuales que se emprenden a lo largo de la ruta. Cuando existen distintas soluciones, es decir distintas rutas que conducen a una misma meta, el costo de cada una puede servir para determinar cuál se elige.

En este caso, el costo de una acción, es decir, de una visita visit(i_j), es el peso asociado al ítem visitado, i_j. Dicho peso se denomina **g(i_j)** y se calcula en función de las preferencias establecidas por el usuario. El cálculo realizado (sección 2.9.2) es tal que el peso de un ítem es menor en cuanto que sus características se adecuan mejor a los gustos del usuario y mayor cuanto más se separan de éstos.

La función costo de ruta se denomina **g(ruta)** y su valor se obtiene mediante la suma de los costos g(i_j) de cada uno de los ítems visitados en ella (ecuación 9).

$$\text{r: visit}(i_1), \text{visit}(i_2),..., \text{visit}(i_n),\ g(r) = g(i_1) + g(i_2) + ... + g(i_n) = \sum_{j=1}^{n} g(i_j) \qquad (9)$$

Dadas dos rutas, r1 y r2, que permiten al usuario lograr su meta, es decir, r1:$s_o \rightarrow s_f$ y r2:$s_o \rightarrow s_g$ tal que prueba-meta(s_f, M') = prueba-meta(s_g, M') = *true*, se prefiere la ruta r1 sobre la ruta r2 si su costo es menor, esto es, g(r1) < g(r2).

2.9.2 Costo de un ítem

El costo de una ruta depende del costo de los ítems que la componen. De forma similar, el costo de un ítem, depende del costo asociado a éste para cada una de sus características. El costo asociado a una característica, está en función de que el valor que ésta presenta en el ítem guste más o menos al usuario.

Dado el conjunto **A** = {idioma, medio, rol, nivel-de-dificultad, autor, fecha-de-creación} que contiene los seis atributos considerados en las preferencias del usuario (preferencias-sobre-ítems), un ítem concreto i_j tiene el costo que se expresa en la ecuación 10.

$$g(i_j) = 1 - \sum_{i=1}^{6} w'(at^i) \times g(i_j, at^i)\text{, donde:} \qquad (10)$$

a) at^i es el i-ésimo atributo incluido en el conjunto A,

b) w'(at^i) es el peso asignado al atributo at^i, y

c) g(i_j, at^i) es la bondad del ítem i_j de acuerdo al atributo at^i.

Tal y como se explicó en el capítulo 12 (Modelo de usuario), el peso **$w'(at^i)$** asociado a la característica at^i se calcula a partir del peso establecido por el usuario para dicho atributo, $w(at^i)$, teniendo en cuenta además su grado de preferencia por las rutas cortas, esto es el peso w(ruta-corta) (véase la ecuación 11).

$$w'(at^i) = w(at^i) \times (1- w(\text{ruta-corta})) \qquad (11)$$

De acuerdo también a lo explicado en el capítulo 12, la bondad del ítem i_j para el atributo at^i, **$g(i_j, at^i)$**, se calcula a partir de la lista de gustos/contra-gustos establecida por el autor para dicha característica, como sigue:

i) Si el valor *v* que presenta la característica at^i en el ítem i_j no aparece en la lista de preferencias del usuario, la bondad del ítem es cero.

   $g(i_j, at^i) = 0$.

ii) En otro caso, siendo p_v la posición de dicho valor en la lista y p_f la última posición de la lista, la bondad es:

   $g(i_j, at^i) = \dfrac{p_f - (p_v - 1)}{p_f}$ si el valor *v* gusta al usuario (bondad positiva).

   $g(i_j, at^i) = - \dfrac{p_f - (p_v - 1)}{p_f}$ si el valor *v* disgusta al usuario (bondad negativa).

Usando las funciones de costo propuestas en 9 y 10, se tienen en cuenta las preferencias del usuario respecto a las características de los ítems que componen la ruta (preferencias-sobre-ítems) y respecto a la longitud de la misma (ruta-corta).

Cuando el peso asociado por el usuario a la longitud de ruta es 1, w(ruta-corta) = 1, siguiendo la ecuación 11 el peso $w'(at^i)$ es cero para cualquier característica (i = 1. .6), por lo tanto la sumatoria $\sum w'(at^i) \times g(i_j, at^i)$ en la ecuación 10 es igual a cero y el coste de cualquier ítem es 1, esto es $\forall i_j \; g(i_j) = 1 - 0 = 1$. En este caso, el coste de una ruta coincide con el número de visitas que la componen, lo cual hace que se elija siempre la ruta más corta ignorando el resto de factores.

En otro caso, w(ruta-corta) < 1, el coste de una ruta depende no sólo del número sino también de las características de los ítems que en ella se visitan. Siguiendo la ecuación 10, el coste de visitar un ítem es igual a:

- 1 – (1– w(ruta-corta)) = w(ruta-corta), en el mejor de los casos.
- 1 + (1– w(ruta-corta)) = 2 - w(ruta-corta), en el caso más desfavorable.

Donde *1-w(ruta-corta)* representa el valor máximo en la sumatoria y lo obtiene un ítem que tiene el mejor valor de bondad, o sea 1, en todas sus características, esto es, $\forall_{i=1..6} g(i_j, at^i) = 1$. Y *–(1-w(ruta-corta))* es el valor mínimo en la sumatoria y lo obtiene un ítem que tiene el peor valor, -1, en todas sus características, esto es, $\forall_{i=1..6} g(i_j, at^i) = -1$.

Para evitar que, aún en el mejor caso, el costo de visitar un ítem sea 0, se exige que el peso de ruta corta sea siempre mayor que este valor, w(ruta-corta) > 0. De esta forma, si por ejemplo, w(ruta-corta) es igual a 0,05, este es el costo asociado a un ítem cuyo valor

para cada atributo coincide con él que más gusta al usuario. En consecuencia, siempre el costo de un ítem es estrictamente positivo y menor que 2, esto es, el valor de $g(i_j)$ pertenece al intervalo abierto (0,2).

Utilizando esta función de costo, y siempre que el peso de ruta-corta no sea uno, un mayor número de ítems en una ruta puede compensarse con la idoneidad de estos, permitiendo elegir una ruta que no es la más corta, pero sí la más apetecible para el usuario.

3. ÁRBOL DE BÚSQUEDA

3.1 En busca de la ruta

Una vez definido formalmente el problema y su solución, en esta sección esbozamos el mecanismo que debe utilizar el sistema para generar la secuencia de visitas que constituyen la ruta finalmente proporcionada al usuario.

El primer paso es **evaluar el estado inicial** del usuario, s_o, y determinar si se trata de un estado meta, esto es siendo M' la meta del usuario se satisface que prueba-meta(s_o, M') = *true*.

En caso de que el estado evaluado, s_{actual}, no satisfaga la prueba de meta, prueba-meta(s_{actual}, M') = *false*, es necesario evaluar otros estados, concretamente aquellos a los que es posible llegar desde éste. Para ello, el sistema aplica los operadores disponibles sobre el estado actual, generando un nuevo conjunto de estados, tal y como se formula en la función subsecuente **$S(s_{actual})$**.

Si tras este proceso de expansión sólo hay **una posibilidad**, es decir existe un único estado en el conjunto generado por $S(s_{actual})$, basta con optar por ella y continuar la búsqueda a partir de dicho estado.

Cuando las **posibilidades son múltiples**, es decir existen dos o más estados en el conjunto generado por $S(s_{actual})$, hay que decidir cuál es el que más conviene y seguir por él. En esto, precisamente, radica la **búsqueda**: escoger una opción, haciendo a un lado las demás para considerarlas posteriormente en caso de ser necesario.

Cuando la expansión del estado actual, $S(s_{actual})$, produce un conjunto vacío, esto es, no existe ninguna posibilidad, es necesario volver atrás y reconsiderar alguna de las posibilidades descartadas en el camino, deshaciendo cuantos pasos en la ruta sean necesarios.

De acuerdo a lo explicado, el sistema sigue eligiendo, probando y expandiendo, hasta dar con una solución adecuada al problema o concluir que esta solución no es posible. La elección del estado que se va a expandir primero o cuál se reconsidera al volver atrás son decisiones que se realizan de acuerdo a una estrategia de búsqueda.

A lo largo del presente capítulo se proponen y comparan distintas estrategias. No obstante, dejando a un lado la estrategia empleada para la búsqueda, el procedimiento general que se sigue es el que se detalla en el algoritmo 1.

Paso 1. Inicialización

Inicializar el estado actual con el estado inicial del usuario: $s_{actual} = s_o$

Inicializar la ruta: ruta = {}

Inicializar un conjunto con los estados a evaluar: candidatos = ∅

Inicializar un conjunto con los estados descartados: descartados = ∅

Paso 2. Probar si el estado actual satisface la meta

Si prueba-meta(s_{actual}, M') = *true* Terminar y devolver la ruta.

En otro caso seguir al Paso 3.

Paso 3. Añadir el estado actual a la ruta y generar los estados que pueden seguir a éste.

ruta = ruta + {s_{actual}}

candidatos = S(s_{actual})

Si candidatos = ∅ ir al Paso 5, en otro caso ir al Paso 4.

Paso 4. Elegir el siguiente estado de entre los estados candidatos.

s_{actual} = elegir (candidatos, estrategiaBúsqueda)

descartados = descartados ∪ (candidatos – s_{actual})

Ir al Paso 2.

Paso 5. Elegir el siguiente estado de entre los estados descartados.

Si descartados = ∅ Terminar y devolver "ruta imposible"

En otro caso:

ruta = ruta - {s_{actual}}

s_{actual} = elegir (descartados, estrategiaBúsqueda)

Ir al Paso 2.

Terminar.

Algoritmo 1. En busca de la ruta

3.2 Construcción del árbol

Para facilitar la comprensión, conviene concebir el proceso de búsqueda como la construcción de un árbol sobrepuesto al espacio de estados. El nodo raíz del **árbol de búsqueda** representa el estado inicial s_o y sus nodos hoja corresponden a los estados que no tienen ningún sucesor en el árbol porque todavía no han sido expandidos.

Son muchas las formas en que pueden representarse los nodos del árbol de búsqueda. Por ejemplo, un **nodo n_k** puede ser representado mediante una estructura de datos con cinco componentes:

1) el estado del espacio de estados al que corresponde, esto es s_k,

2) el nodo que lo generó, denominado nodo padre,

3) el operador que se aplicó para obtenerlo,

4) la profundidad del nodo, o sea el número de nodos desde la raíz, n_o, hasta dicho nodo, y

5) el costo de la ruta que va en el árbol desde el estado inicial s_o hasta él.

Sin embargo, por simplicidad, en los ejemplos que presentamos a continuación, representamos un nodo n_k únicamente con el estado al que corresponde (1), esto es s_k. Y etiquetamos la rama que une el nodo n_k a su nodo padre, n_x, con el operador que lo generó (2 y 3). Puesto que todos los operadores son de tipo visit(i_j), lo único que se anota es el nombre del ítem visitado, i_j, el cuál determina unívocamente la regla de actualización aplicada, esto es $Ru(i_j)[s_x] = s_k$.

Supongamos el ejemplo de la figura 1, donde se muestra el conjunto de reglas de conocimiento y actualización (figura 1.1) asociadas a los ítems I1, I2,..., I7 de la estructura de navegación que se adjunta (figura 1.2).

Supongamos, además, que el usuario se encuentra en un estado de desconocimiento absoluto, esto es s_o= (0, 0, 0, 0, 0, 0, 0).

I1	Rk(I1) = ∅
I2	Rk(I2) = ∅
I3	Rk¹(I3): K(I1) ≥ 1 and‴ true→ Visitar(I3) Rk²(I3): K(I2) ≥ 2 and‴ K(I2) ≤ 2 → Visitar(I3) Rk³(I3): K(I2) ≥ 3 and K(I4) ≥ 4 and‴ true → Visitar(I3) Rk⁴(I3): K(I6) ≥ 3 and K(I4) ≥ 4 and‴ true → Visitar(I3)
I4	Rk¹(I4): true and‴ K(I5) ≤ 3 and K(I2) ≤ 2 → Visitar(I4)
I5	Rk¹(I5): K(I4) ≥ 4 and‴ true → Visitar(I5)
I6	Rk¹(I6): K(I4) ≥ 4 and‴ K(I2) ≤ 2 → Visitar(I6)
I7	Rk¹(I7): K(I4) ≥ 4 and‴ true → Visitar(I7)

Ru(I1)	Si Visitado(I1) entonces Fix-abs(I1, "total");
Ru(I2)	Si Visitado(I2) entonces Fix-abs(I2, "total");
Ru(I3)	Si Visitado(I3) entonces Fix-abs(I3, "total");
Ru(I4)	Si Visitado(I4) entonces Fix-abs(I4, "total"), Inc-abs(I6,1), Inc-abs(I7,2);
Ru(I5)	Si Visitado(I5) entonces Inc-abs(I5, 2), Fix-rel(I1);
Ru(I6)	Si Visitado(I6) entonces Fix-abs(I6, "total"), Inc-abs(I2,1);
Ru(I7)	Si Visitado(I7) entonces Fix-abs(I7, "total");

Figura 1.1: Un ejemplo – Rk y Ru

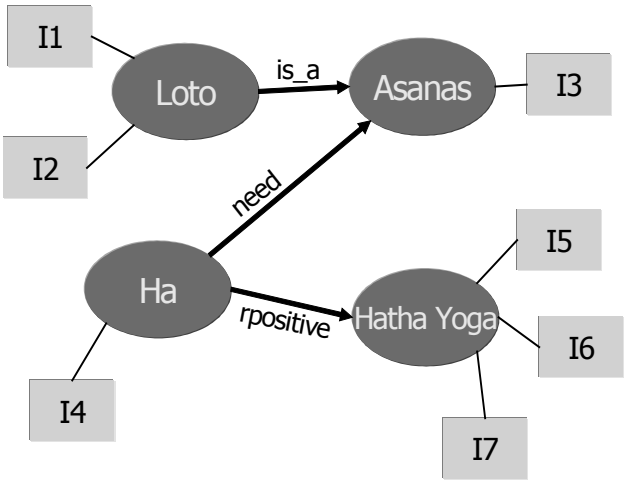


Figura 1.2: Un ejemplo - EC_p

El árbol de búsqueda representado parcialmente en la figura 2 representa todos los posibles estados de conocimiento que este usuario puede alcanzar con la navegación de la estructura proporcionada a partir de su estado inicial.

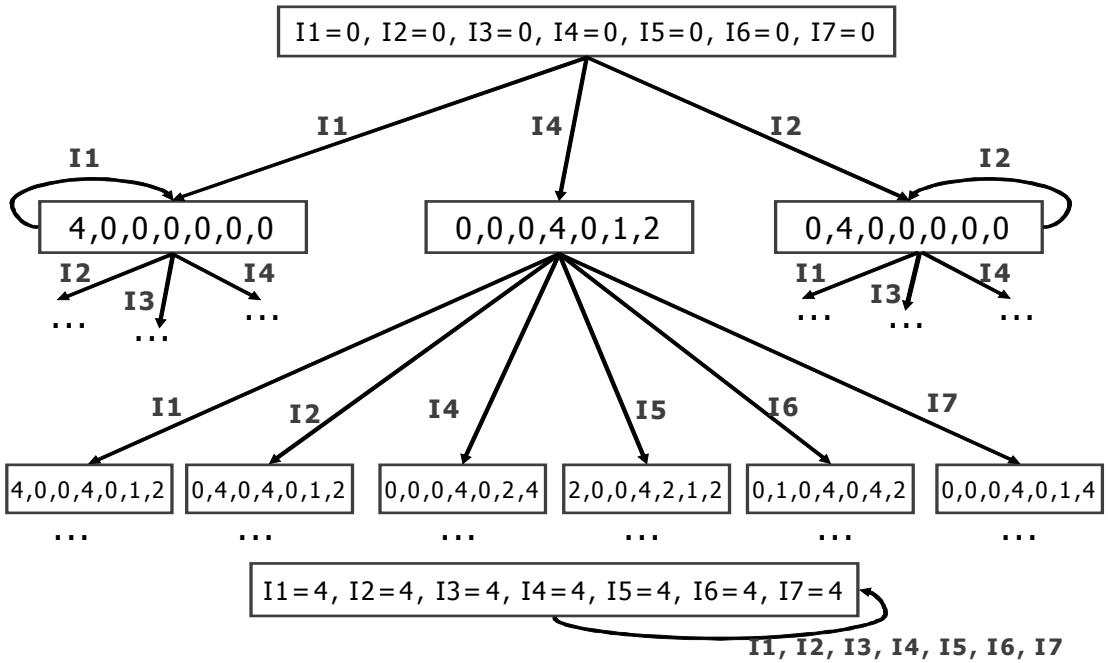


Figura 2: Árbol de búsqueda para el ejemplo de la figura 1

Observe que cada nodo del árbol expresa en valor numérico, el grado de conocimiento que el usuario posee, en un determinado paso de navegación, sobre cada uno de los ítems mostrados en la estructura de navegación actual.

El nodo raíz del árbol, representa el estado inicial del usuario, s_o= (0, 0, 0, 0, 0, 0, 0). En este estado, el grado de conocimiento sobre cada uno de los siete ítems, es "nulo", esto es K(Ij) = 0 para j:1. .7.

A partir del estado de conocimiento s_o, los únicos ítems accesibles son aquellos que no tienen ninguna restricción de accesibilidad. Este es el caso de los ítems I1 e I2 para los

que no se ha definido ninguna regla de conocimiento, y el caso del ítem I4 que tiene una única regla asociada, donde sólo aparecen requisitos de idoneidad.

El conjunto de estados obtenido como resultado de la función de expansión $S(s_o)$ es $\{s_1, s_2, s_4\}$. Los operadores aplicados para generar cada estado del conjunto han sido respectivamente visit(I1), visit(I2) y visit(I4). Cada uno de ellos se ha obtenido ejecutando la regla de actualización del ítem visitado sobre el estado de conocimiento s_o, esto es: $s_1 = Ru(I1)[s_o] = (4,0,0,0,0,0,0)$, $s_2 = Ru(I2)[s_o] = (0,4,0,0,0,0,0)$ y $s_4 = Ru(I4)[s_o] = (0,0,0,4,0,1,2)$.

En consecuencia, tres nodos hijos, correspondientes a los estados s_1, s_2 y s_4, se han generado en el árbol tras la expansión del nodo raíz. Observe, que las ramas que unen estos nodos a la raíz se etiquetan con el identificador del ítem accesible (I1, I2 o I4) que se ha visitado para generarlos.

3.3 Características del árbol

3.3.1 Repetición de estados

Dada la naturaleza de las reglas de conocimiento y actualización, en la mayoría de los casos es posible llegar a un mismo estado de maneras distintas. Esto es, a partir de un estado s_1 pueden existir dos o más rutas que conducen a un estado s_2.

Una cuestión interesante desde el punto de vista de la eficiencia, es evitar perder tiempo expandiendo estados que ya se encontraron y expandieron anteriormente en alguna otra ruta. Para ello es necesario comprobar cada vez que se expande un nodo, n_k, cuáles de los estados s_x obtenidos en la expansión, $s_x \in S(s_k)$, han sido generados y expandidos previamente, y en tal caso no volver a expandirlos.

En nuestro ejemplo, a partir del estado inicial s_o= (0,0,0,0,0,0,0) es posible llegar al estado (4,4,0,0,0,0,0) a través de dos secuencias de visitas: ruta1 = {visit(I1), visit(I2)} y ruta2 = {visit(I2), visit(I1)}.

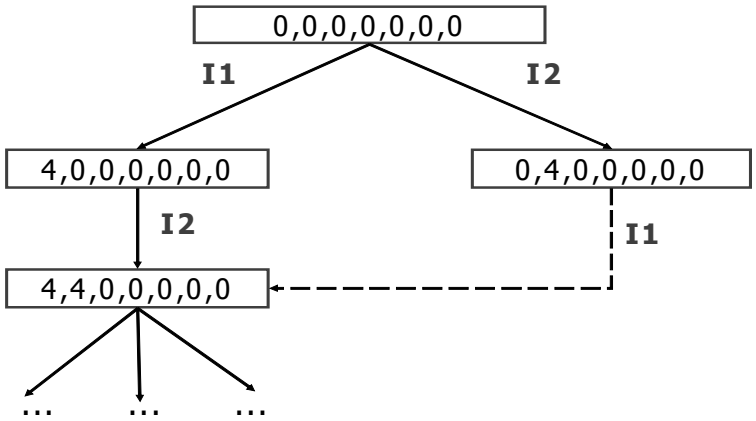


Figura 3.1: Estados repetidos

Si se ha expandido completamente el subárbol que cuelga del nodo (4,4,0,0,0,0,0) para la secuencia I1, I2, no tiene sentido generarlo de nuevo cuando se obtiene ese mismo

nodo tras la secuencia I2, I1. De modo que, como se observa en la figura 3.1, al detectar esta repetición cuando se expande el estado (0,4,0,0,0,0,0), no se genera de nuevo el nodo (4,4,0,0,0,0,0), sólo una rama hasta él (marcada con línea discontinua).

Está demostrado que **evitar la repetición de estados** produce una reducción exponencial del tiempo de búsqueda, sin embargo, no generar ningún estado que se haya generado alguna vez anteriormente, tiene una complejidad espacial derivada de tener que almacenar en memoria todos los estados generados hasta el momento.

El número máximo de estados posibles en el problema que nos ocupa es $O(5^n)$, donde 5 representa los distintos grados de conocimiento posibles sobre un ítem y *n* es el número de ítems en la estructura de navegación actual. Aunque en la práctica, a poco que se defina un conjunto sensato de reglas de conocimiento y actualización, el número de estados se reduce considerablemente, es conveniente plantear otras alternativas con requerimientos de memoria menos exigentes.

Una propuesta ampliamente aceptada en la literatura científica [Russell, 96], consiste en *no crear rutas que contengan ciclos*. Es decir, redefinir la función de expansión (S) para que se niegue a generar sucesores de un nodo idénticos a cualquiera de sus ancestros.

Puesto que las reglas de actualización nunca decrementan el conocimiento del usuario, el único caso en que es posible regresar a un antecesor, es cuando el nodo hijo coincide con su nodo padre. Debiendo únicamente modificar la función de expansión para que no genere los sucesores cuyo estado de conocimiento sea el mismo que el del nodo padre, lo cual supone un esfuerzo computacional despreciable (véase la figura 3.2).

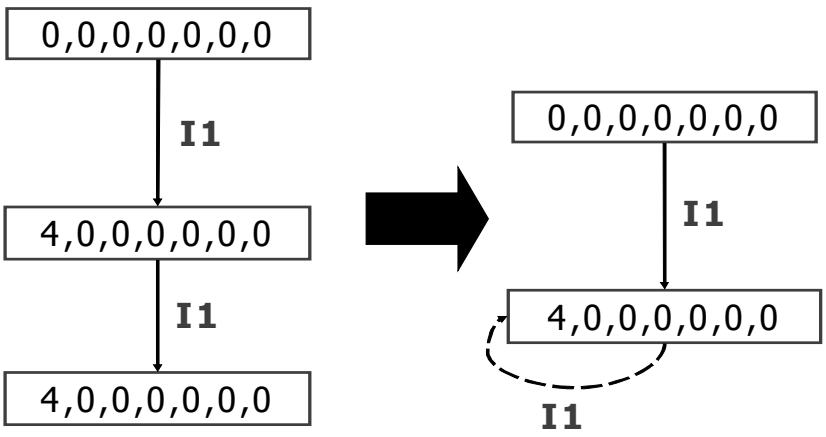


Figura 3.2: Estados repetidos: ciclos I

Aprovechando las particularidades de nuestro problema, se puede añadir, sin demasiado esfuerzo, la siguiente mejora: Cuando la ejecución de una regla de actualización $Ru(i_j)$ sobre un estado s_x produce el mismo estado, dicha regla no puede producir ninguna modificación al aplicarse en los estados sucesores de s_x. Por lo tanto, se puede anticipar la existencia de un ciclo con el operador $visit(i_j)$ en la expansión de todos los estados sucesores de s_x (véase la figura 3.3).

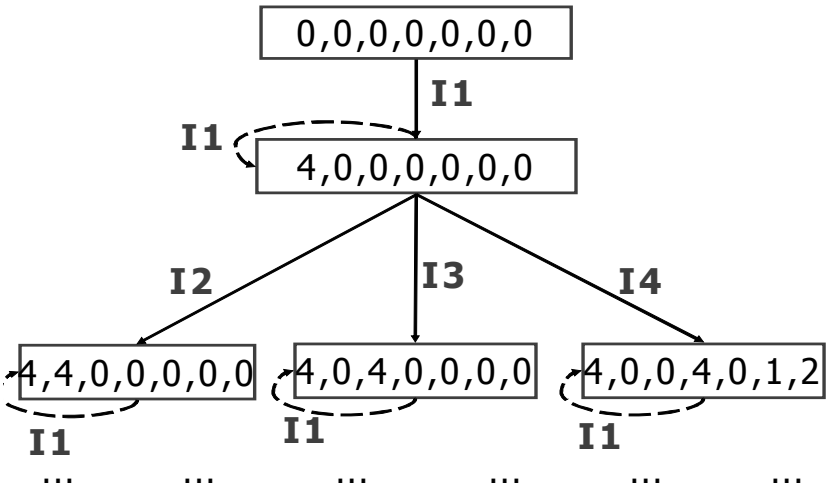


Figura 3.3: Estados repetidos: ciclos II

La afirmación anterior es posible gracias a las siguientes propiedades de los predicados de actualización (capítulo 9) incluidos en la regla $Ru(i_j)$:

a) Cualquier actualización de tipo "1ª-vez", sólo tiene efecto la primera vez que se ejecuta, por lo tanto, si no produce cambios en el estado s_k tampoco los puede producir en ningún sucesor de éste.

b) Una actualización "fija" y "absoluta" sólo puede provocar cambios la primera vez que se ejecuta, por lo tanto, igual que en a), si no tiene efecto en el estado s_k tampoco lo tiene en sus sucesores.

c) Una actualización "incremental" y "absoluta" deja de producir cambios cuando el conocimiento del ítem actualizado ha llegado a "total". Puesto que el conocimiento no decrece, si en el estado s_k no tiene efecto tampoco lo va a tener en sus sucesores.

d) Una actualización "relativa", ya sea fija o incremental, deja de tener efecto cuando el conocimiento del ítem cabeza de la regla ha llegado a "total". Así, por el mismo motivo que en c), si en s_k no origina ningún cambio tampoco en sus estados sucesores.

3.3.2 Solución garantizada

Sea cual sea el conjunto de reglas de actualización y conocimiento a partir de las cuales se genera el árbol de búsqueda, siempre está garantizada la existencia de un nodo n_k, que representa el estado de conocimiento absoluto, es decir s_k = (4, 4, 4,..., 4). Esto es posible gracias a las propiedades de consistencia que se exigen al autor durante la definición de dichas reglas (capítulo 10, Consistencia $Rk \cup Ru$), y que pueden descomponerse en las siguientes premisas:

a) Partiendo de cualquier estado de conocimiento, el conjunto de reglas $Rk(EC_A^j) \cup Ru(EC_A^j)$ definido sobre la presentación EC_P^k asegura que **no existen ítems inalcanzables**. Esto garantiza la posibilidad de llegar a acceder a cualquier ítem $i_j \in I(EC_P^k)$, independientemente de la navegación realizada hasta el momento.

b) Puesto que las reglas de actualización incrementan o mantienen el conocimiento del usuario, cuando un ítem se vuelve accesible lo sigue siendo de ahí en adelante. Esto permite garantizar, a partir de a), que **existe un momento en el que todos los ítems de la estructura de navegación son accesibles** para el usuario.

c) Las reglas de actualización aseguran que la visita repetida a un ítem consigue un conocimiento "total" sobre éste. Por lo tanto, a partir de la premisa b) se deduce que **cualquier usuario puede alcanzar el estado de conocimiento total s_k = (4, 4,..., 4)**. En el momento en que un ítem i_j es accesible para conocimiento totalmente, $K(i_j)$ = "total", basta con visitarlo una vez si su actualización es "fija" o un máximo de cuatro veces si su actualización es "incremental".

Del cumplimiento de la premisa c) es posible concluir que, para cualquier meta M', existe en el árbol de búsqueda un camino que va desde el estado inicial del usuario hasta un estado s_m que satisface prueba-meta(s_m, M'). El razonamiento es inmediato:

- Se garantiza que existe en el árbol un nodo representando el estado s_f = (4, 4,..., 4) y que es posible llegar a él desde cualquier nodo del árbol.

- El estado s_f = (4, 4,..., 4) es un estado meta, ya que satisface la prueba de meta para cualquiera que sea ésta, o sea $\forall$M', prueba-meta(s_f, M') = *true*.

- Por lo tanto, siempre como mínimo existe un estado meta, s_f, y uno o más caminos para llegar hasta él.

El objetivo, es proporcionar al usuario una secuencia de visitas que tras ser realizada le permita alcanzar un estado de conocimiento que satisfaga su meta. Para ello, el sistema va construyendo el árbol de búsqueda hasta dar con una solución. La estrategias de búsqueda que se pueden aplicar son muy diversas, pero en todos los casos **está garantizado que existe solución**. Es decir, siempre hay al menos una ruta, normalmente varias, que conduce al usuario desde su estado de conocimiento actual hasta un estado de conocimiento que satisface la prueba de meta. El hecho de saber de antemano que existe solución resulta muy útil en la ejecución de algunas de estas estrategias (véase la sección 4).

3.3.3 Nodos hoja

En nuestro problema cualquier nodo n_k del árbol genera en su expansión un conjunto de sucesores no vacío, esto es, existe al menos un estado en el conjunto $S(s_k)$. Esto se debe a que:

a) En el estado inicial, s_o, existe al menos un ítem accesible, en otro caso no se podría garantizar la alcanzabilidad de todos los ítems.

$$S(s_o) \neq \varnothing \qquad (12.1)$$

b) Debido a la naturaleza de las reglas de actualización, el estado de conocimiento representado en un nodo s_2 no puede ser nunca menor que el estado de conocimiento representado en el nodo s_1 que lo generó (véase la ecuación 12.2).

$$s_1 = (K(i_1)^1, K(i_2)^1, \ldots, K(i_n)^1), \quad s_2 = (K(i_1)^2, K(i_2)^2, \ldots, K(i_n)^2)$$

$$\forall_j \text{ si } s_2 = Ru(i_j)[s_1] \text{ entonces } \forall_{k=1..n} \; K(i_k)^2 \geq K(i_k)^1 \qquad (12.2)$$

c) Las reglas de conocimiento son tales que si un estado de conocimiento s_2 es igual o mayor que un estado de conocimiento s_1, el conjunto de ítems accesibles en s_2 contiene todos los ítems accesibles en s_1 y posiblemente alguno o algunos más (véase la ecuación 12.3).

$$s_1 = (K(i_1)^1, K(i_2)^1, \ldots, K(i_n)^1) \text{ y } s_2 = (K(i_1)^2, K(i_2)^2, \ldots, K(i_n)^2),$$

$$\text{si } \forall_{k=1..n} K(i_k)^2 \geq K(i_k)^1 \text{ entonces: } S(s_1) \subseteq S(s_2) \qquad (12.3)$$

Conforme a lo anterior y de forma estricta, en el árbol de búsqueda sólo son nodos hoja aquellos pendientes de expandir. Ya que **no existe ningún nodo que al expandirlo no genere ningún nodo hijo**. Sin embargo, esto no significa, que el árbol sea infinito, ya que muchos de los nodos generados en la expansión son nodos repetidos, incluso idénticos al nodo que los generó (ver sección 3.3.1).

No obstante, puesto que desde cualquier nodo n_k es posible llegar al nodo (4, 4,..., 4), la expansión del estado representado en n_k, $S(s_k)$, debe incluir, forzosamente, al menos un estado distinto al representado en n_k, salvo que dicho estado sea precisamente el de conocimiento absoluto. Esto es $\exists s_x \in S(s_k)$ tal que $s_x \neq s_k$ o $s_k = (4, 4, ..., 4)$.

A medida que se construye el árbol de búsqueda, se generan mayor número de estados repetidos, hasta llegar al nodo (4, 4,..., 4). Este nodo genera en su expansión tantos hijos como ítems existen en la estructura de navegación actual (todos los ítems son ahora accesibles). Sin embargo, todos los hijos generados son idénticos a su padre. Este hecho permite considerar el nodo que contiene el **estado de conocimiento "total" como el único nodo hoja del árbol**, ya que sea cual sea el operador aplicado, o sea el ítem visitado, el usuario se mantiene en ese nodo.

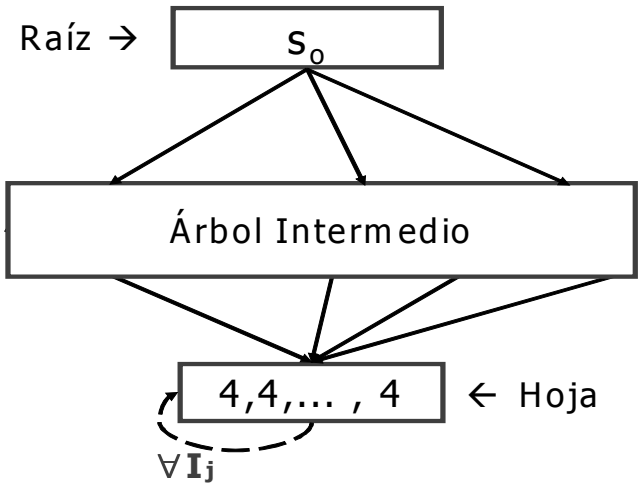


Figura 4: Estructura general de un árbol de búsqueda

Como se muestra en la figura 4, en nuestro problema, cualquier árbol de búsqueda representa en el nodo raíz el estado de conocimiento inicial del usuario, y en el nodo hoja el estado de conocimiento absoluto. De forma que entre ellos, existen múltiples nodos y conexiones, que dependen del conjunto concreto de reglas de conocimiento y actualización.

4. ESTRATEGIAS DE BÚSQUEDA

Una vez descrita la estructura general del árbol de búsqueda, el tratamiento de los estados repetidos y la garantía de una solución, entramos de lleno en la propuesta y comparación de distintas estrategias de búsqueda. En concreto se va a evaluar cada estrategia en función de los cuatro criterios que se mencionan a continuación [Russell, 96]:

- **Completitud**: ¿La estrategia garantiza encontrar una solución?
- **Complejidad temporal**: ¿Cuánto tiempo se necesita para encontrar la solución?
- **Complejidad espacial**: ¿Cuánta memoria se necesita para efectuar la búsqueda?
- **Optimización**: ¿Con esta estrategia se encuentra la solución de más alta calidad en el caso de que existan varias soluciones?

Las estrategias evaluadas son las que se resumen en la siguiente tabla, agrupadas en dos categorías: búsqueda ciega y búsqueda respaldada con información.

Tabla 1: Estrategias de búsqueda

Búsqueda ciega	Búsqueda preferente por amplitud
	Búsqueda preferente por profundidad
	Búsqueda por profundización iterativa
	Búsqueda bidireccional
	Búsqueda de costo uniforme
Búsqueda con información	Búsqueda avara
	Búsqueda A*

4.1 Búsqueda preferente por amplitud

La búsqueda preferente por amplitud es un tipo de **búsqueda ciega** donde no se tiene información acerca de la cantidad de pasos necesarios, o sobre el costo de la ruta, para pasar del estado actual a un estado que satisface la prueba de meta. Lo único que permite es diferenciar entre un estado meta y otro que no lo es.

El funcionamiento es muy sencillo: primero se expande el nodo raíz y luego todos los nodos generados por éste; después, sus sucesores, y así sucesivamente. En general, todos **los nodos que están en la profundidad *d* del árbol se expanden antes que los nodos que estén en la profundidad *d+1***. Es pues, una búsqueda muy sistemática, que primero toma en cuenta todas las rutas de longitud 1, luego las de longitud 2, etc (véase la figura 5).

Esta estrategia de búsqueda es *completa* porque con suficiente tiempo y memoria, siempre encuentra una solución. Si son varias las soluciones, encuentra primero la ruta más corta, es decir la de menor número de visitas. Sin embargo, *no* asegura que la ruta obtenida sea *óptima* puesto que no tiene en cuenta las preferencias del usuario.

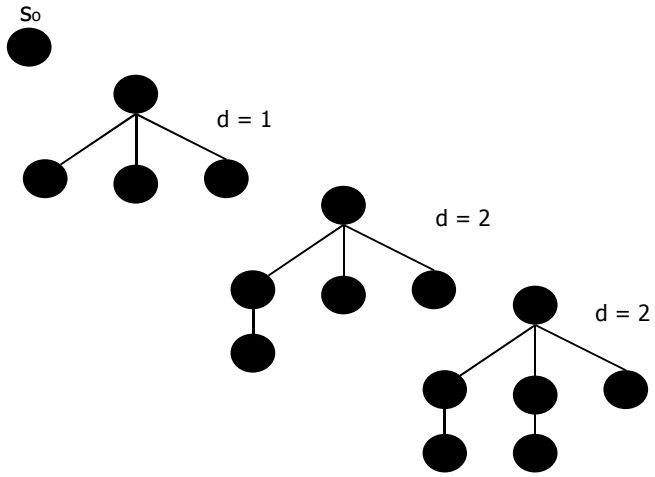


Figura 5: Búsqueda primero en amplitud

La *complejidad espacial y temporal* teórica es $O(n^{4n})$ [Russell, 96], donde *n* es el número de ítems que se muestran en la estructura de navegación actual. Esta cota se obtiene, al considerar en el peor de los casos, un factor de ramificación *n* y una longitud de ruta $4 \times n$. Este peor caso es definido por las siguientes hipótesis:

1) El estado inicial del usuario es el estado de desconocimiento absoluto, s_o= (0,0,..., 0).

2) El único estado que satisface la prueba de meta es el estado de conocimiento total s_m= (4,4,..., 4).

3) No existen restricciones de accesibilidad para ningún ítem, por lo tanto, en cada nodo del árbol de búsqueda los *n* ítems de la estructura de navegación son accesibles.

4) Todas las reglas de actualización actualizan únicamente el ítem visitado, mediante una actualización incremental de un único grado. Esto es, $\forall i_j$ Ru(i_j): Si visitado(i_j) entonces Inc-abs(i_j, 1);. Por lo tanto son necesarias cuatro visitas a un ítem para conocerlo totalmente, y $4 \times n$ para conocer los *n* ítems con grado "total".

La complejidad anterior se obtiene a partir del siguiente razonamiento: La raíz del árbol de búsqueda genera *n* nodos en el primer nivel, cada uno de los cuales genera *n* nodos más, con un total de n^2 nodos en el segundo nivel. De manera que si existe una ruta de longitud *4n*, la cantidad máxima de nodos expandidos antes de poder encontrar esa solución es $1 + n + n^2 + n^3 + ... + n^{4n}$, lo cual constituye un orden de complejidad $O(n^{4n})$.

Sin embargo, sabemos que dentro de esos nodos existe una fuerte repetición, donde a menudo el nodo generado coincide exactamente con el nodo que lo generó. Si eliminamos estas repeticiones (sección 3.3.1), el número máximo de nodos es 5^n. Este valor contempla todas las combinaciones distintas con *n* elementos y cinco valores posibles para cada uno, y permite reducir el orden de complejidad para el peor caso a $O(5^n)$, aún exponencial.

No obstante, esta cota exponencial no es determinante, ya que a poco que se mejoren las condiciones impuestas en el peor caso, la complejidad temporal y espacial se reduce enormemente. Y es que, aunque las dos primeras hipótesis son factibles, las dos últimas son bastante improbables.

Normalmente el autor define un conjunto de reglas de conocimiento que permite dirigir al usuario durante su navegación por conocimiento, evitando que acceda al contenido de un ítem hasta que conozca con cierto nivel otros ítems necesarios para comprenderlo. Estas **restricciones de accesibilidad permiten podar el árbol** de búsqueda, a menudo drásticamente, de forma que se reduce el factor de ramificación (hipótesis 3).

Por otro lado, lo habitual es que las reglas de actualización definan incrementos de conocimiento más generosos sobre el ítem visitado, otorgando conocimiento "total" sobre éste después de una o dos visitas a lo sumo. Además, las reglas de actualización suelen implicar actualizaciones sobre otros ítems relacionados, lo que junto a lo anterior reduce considerablemente la longitud de la ruta más larga (hipótesis 4).

De esta forma, si calculamos la complejidad en un caso más favorable, aún manteniendo las hipótesis 1 y 2 del peor caso, obtenemos un orden muy alentador de *O(n)*. Para ello, hemos supuesto las siguientes hipótesis:

4) Todas las reglas de actualización fijan el ítem visitado al grado de conocimiento "total". Esto es, $\forall i_j$ Ru(i_j): Si visitado(i_j) entonces Fix-abs(i_j, 4);. Por lo tanto, sólo es necesaria una visita a un ítem para conocerlo totalmente, reduciéndose a *n* la longitud de la ruta más larga.

3) Las restricciones de accesibilidad son tales que para cada nodo del árbol su expansión genera un único nodo hijo distinto de si mismo, esto es, el factor de ramificación es 1.

La situación descrita en 3) ocurre, por ejemplo, cuando los ítems se hacen accesibles de uno en uno, de modo que en un nodo son accesibles los mismos ítems que en su nodo padre más uno. Con las reglas de actualización definidas en 4) la visita a los ítems previamente accesibles produce un ciclo sobre el nodo actual, de modo que sólo el nuevo ítem accesible genera un nodo no repetido.

La complejidad se obtiene a partir del siguiente razonamiento: La raíz del árbol de búsqueda genera al expandirse *1* nodo obtenido tras la visita del único ítem accesible inicialmente, i_1. Según la hipótesis 3, en dicho nodo hay, ahora, dos ítems accesibles, i_1 e i_2. Por lo que se generan *2* nodos en el segundo nivel del árbol, donde, uno de ellos, el generado por el operador visit(i_1), es idéntico a su padre. En el nivel tres sólo se expande el nuevo nodo, que da lugar a *3* nodos a través de la visita de los ítems accesibles i_1, i_2 e i_3, donde sólo el nodo generado mediante el operador visit(i_3) es distinto del nodo actual, etc. Siguiendo la hipótesis 4, si la ruta más larga tiene longitud *n*, la cantidad máxima de nodos expandidos antes de poder encontrar la solución es $1 + 2 + 3 + 4 + ... + (n-1) + n$, lo que constituye un orden de complejidad O(n).

Como realmente, sólo se genera un nodo distinto en cada nivel del árbol, el número total de nodos distintos es exactamente *n*, descartando un total de $1 + 2 + 3 + 4 + ... + (n-1)$ nodos repetidos. Además, en este caso, es inmediato detectar la repetición porque se trata de ciclos sobre un mismo nodo.

Un ejemplo de la situación explicada se muestra en la figura 6, donde se aplican las reglas de actualización por defecto y las reglas de conocimiento exigen al usuario tener un conocimiento mayor que "nulo" sobre el ítem i_j para poder visitar el ítem i_{j+1}. Siendo por tanto necesario: conocer I1 antes de visitar I2, conocer I2 antes de visitar I3, conocer I3 antes de visitar I4, etc.

Observe que el árbol obtenido es realmente pequeño, ya que detectando los ciclos sobre un mismo estado, tal y como se ha explicado anteriormente, quedan tan sólo 7 nodos en el árbol.

I1	Rk(I1) = ∅
I2	Rk¹(I2): K(I1)>= 1 → Visitar(I2)
I3	Rk¹(I3): K(I2)>= 1 → Visitar(I3)
I4	Rk¹(I4): K(I3)>= 1 → Visitar(I4)
I5	Rk¹(I5): K(I4) >= 1 → Visitar(I5)
I6	Rk¹(I6): K(I5) >= 1 → Visitar(I6)
I7	Rk¹(I7): K(I6) >= 1 → Visitar(I7)

I1	Ru(I1):Si visitado(I1) entonces Fix-abs(I1, "total");
I2	Ru(I2):Si visitado(I2) entonces Fix-abs(I2, "total");
I3	Ru(I3):Si visitado(I3) entonces Fix-abs(I3, "total");
I4	Ru(I4):Si visitado(I4) entonces Fix-abs(I4, "total")
I5	Ru(I5):Si visitado(I5) entonces Fix-abs(I5, "total");
I6	Ru(I6):Si visitado(I6) entonces Fix-abs(I6, "total");
I7	Ru(I7):Si visitado(I7) entonces Fix-abs(I7, "total");

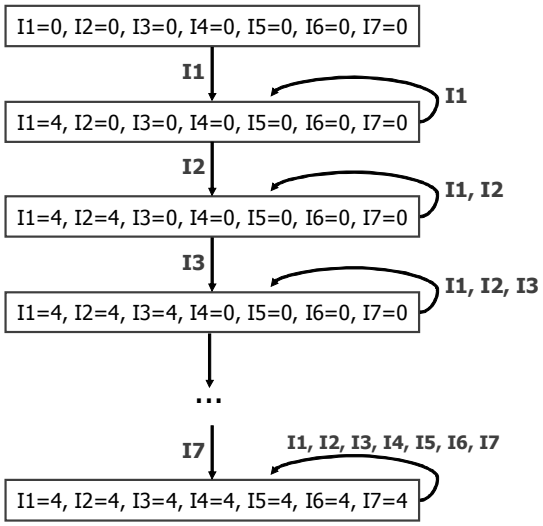


Figura 6: Siete nodos en el árbol para n = 7

Compruebe además, que el caso explicado es bastante favorable, pero no el mejor posible, ya que el árbol de la figura, quedaría reducido a tan sólo dos nodos, el raíz (0,0,..., 0) y el hoja (4,4,..., 4), si la regla de actualización del ítem I1 fijase a "total"el grado de conocimiento del usuario sobre cada uno de los ítems de la estructura de navegación.

Es decir, si en lugar de la regla de actualización por defecto, se tuviese Ru(I1): Si visitado(I1) entonces Fix-abs(I1,"total"), Fix-abs(I2, "total"), Fix-abs(I3, "total"), Fix-

abs(I4, ”total”), Fix-abs(I5, ”total”), Fix-abs(I6, ”total”), Fix-abs(I7, ”total”);, el orden sería O(1). Obviamente se trata de una situación tan improbable como la del peor caso.

Es indudable el importante papel que juega el conjunto de reglas de conocimiento y actualización a la hora de reducir el número de nodos en el árbol de búsqueda. A modo ilustrativo, se ha mostrado la gran diferencia que existe en el orden de complejidad para el mejor, O(1), y peor caso, $O(n^{4n})$, con la estrategia de búsqueda considerada.

Obviamente, el caso medio va a presentar un orden intermedio, ni tan favorable como O(1) ni tan tremendo como $O(n^{4n})$. Este orden, cabe esperar que sea menor cuanto más restrictivo sea el conjunto de reglas de conocimiento y más generoso el conjunto de reglas de actualización. Esto se debe a que, las restricciones de acceso reducen el factor de ramificación en los primeros niveles del árbol, y las **actualizaciones de conocimiento sustanciosas acortan la longitud de la ruta** hasta el estado de conocimiento absoluto.

Se propone como trabajo futuro un estudio estadístico y experimental, que permita demostrar que a poco que se mejore las hipótesis del peor caso, la búsqueda en amplitud encuentra la ruta más corta en un tiempo y con un consumo de recursos razonable. Para los experimentos, se debe suponer un número de ítems en torno a la veintena. Ya que, para evitar problemas de sobrecarga cognitiva, desorientación y falta de comprensión, no debería haber muchos más en una estructura de navegación.

En cualquier caso, el sistema puede ofrecer la posibilidad de utilizar este tipo de búsqueda, de modo que sea el propio usuario quien establezca el **tiempo máximo que puede esperar** para obtener la ruta. Así, si el usuario no para la búsqueda antes, ni se encuentra una solución, ésta se detiene una vez agotado el límite de tiempo. Si dentro de tiempo el sistema encuentra la ruta, se la proporciona al usuario, en otro caso se le aconseja repetir la búsqueda con otra estrategia más apropiada.

4.2 Búsqueda preferente por profundidad

La búsqueda preferente por profundidad es, al igual que la anterior, una estrategia de **búsqueda ciega**. En la búsqueda por profundidad [Nilsson, 87] **siempre se expande uno de los nodos que se encuentran en lo más profundo del árbol** (véase la figura 7).

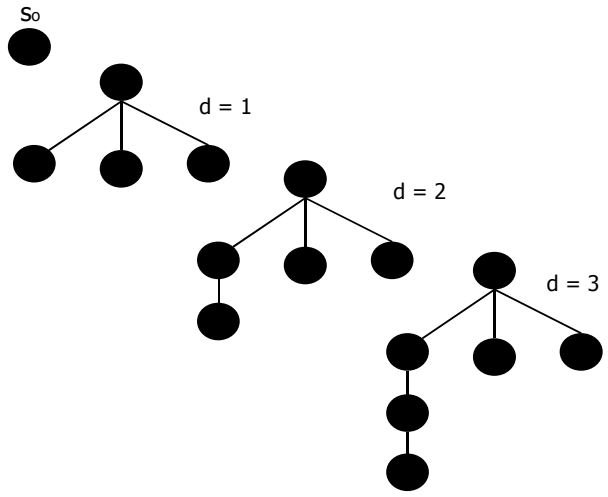


Figura 7: Búsqueda primero en profundidad

Normalmente, sólo se revierte la búsqueda y se expanden los nodos de niveles menos profundos cuando se llega a un nodo que no es meta y no tiene expansión. Sin embargo, en nuestro problema es imposible que tenga lugar esta situación de "callejón sin salida".

Esto se debe a que como se explicó en las secciones 3.3.2 y 3.3.3, todos los nodos producen en su expansión al menos un nodo distinto de si mismo, excepto el nodo final s_f = (4,4,...,4), y está garantizado que desde cualquier nodo del árbol representando un estado $s_i \geq s_{o,}$ existe una secuencia de visitas hasta un nodo meta $s_m \leq s_f$.

En consecuencia, si en cada paso de la búsqueda se opta por el primer estado distinto de del estado expandido, tarde o temprano, dicho estado va a satisfacer la prueba de meta, ya que éste va creciendo en cada paso hasta convertirse en el estado de conocimiento "total", si no satisface la meta antes. Por lo tanto, en nuestro problema conocemos de antemano que **la primera búsqueda en profundidad va a encontrar una solución sin necesidad de vuelta atrás**.

Es más, se garantiza que *se consigue una solución como mucho en 4×n pasos* cuando la ruta encontrada se corresponde con la ruta de peor caso (hay que visitar cuatro veces cada ítem para pasar de conocimiento "nulo" a conocimiento "total"). Cabe esperar que, en un caso medio, el número de pasos sea *n* o menor, lo que significa alcanzar la meta visitando una vez cada ítem o incluso sin haberlos visitado todos.

Retomando el ejemplo de la figura 1, suponemos un usuario que únicamente ha visitado el ítem I1, y desea alcanzar la meta de conocimiento M' = {(I4, "high"), (I7, "low")}.

Este usuario se encuentra en posesión del estado de conocimiento s_i = (4, 0, 0, 0, 0, 0, 0), que pasa a ser el nodo raíz del árbol de búsqueda. A partir de éste, el sistema debe encontrar una ruta con destino en un estado s_m que satisfaga la prueba-meta(s_m, M'). Esto es, s_m = (_, _, _, v4, _, _, v7) con v4 ≥ 3 ("high") y v7 ≥ 1 ("low").

En la figura 8 se muestra la búsqueda realizada en profundidad para obtener la ruta.

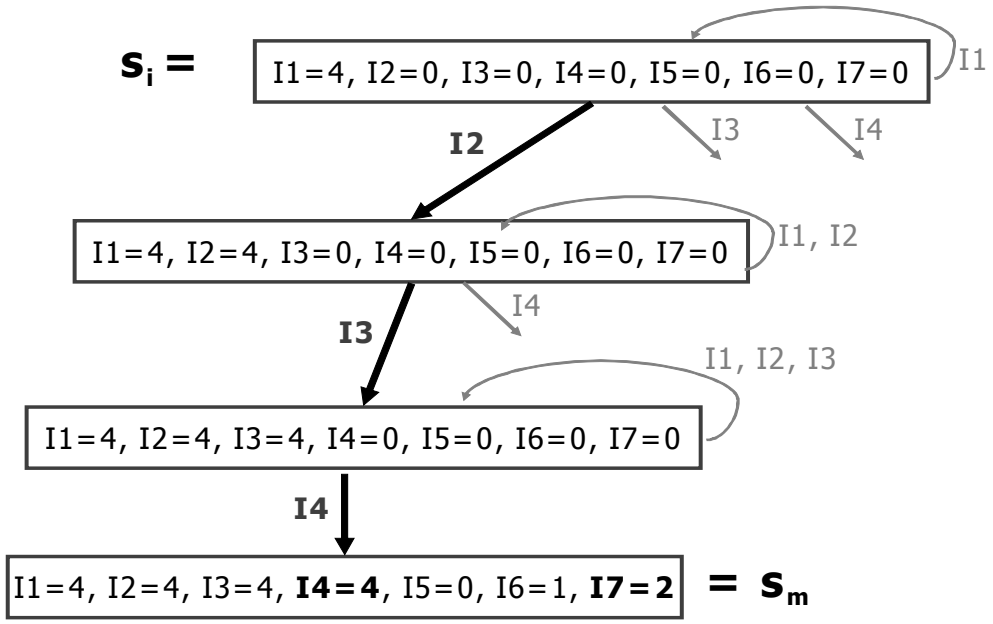


Figura 8: Búsqueda en profundidad

Observe, que la ruta encontrada es: r = {visit(I2), visit(I3), visit(I4)} y el destino de ésta el estado s_m = (4,4,4,**4**,0,1,**2**), con v4 = 4 > 3 y v7 = 2 > 1.

En nuestro problema, la búsqueda preferente por profundidad va a ser normalmente más eficiente que la búsqueda en amplitud. Esto se debe a que desciende en el árbol, siempre de un estado a otro de mayor conocimiento. Mientras que la búsqueda en amplitud se desplaza a lo ancho, de un estado a otro que puede ser mayor pero también menor.

En el árbol de búsqueda, los nodos más profundos representan estados de conocimiento mayores que permiten satisfacer metas imposibles para los niveles superiores. No obstante, aunque los estados situados en los niveles menos profundos del árbol tienen menos expectativas de satisfacer una prueba de meta, dependiendo de la meta, es posible que esto ocurra, obteniéndose entonces muy buenos resultados con la búsqueda en anchura que procura la ruta más corta.

Durante la búsqueda en profundidad la obtención rápida de resultados está garantiza independientemente de la meta, ya que en el peor caso se baja en la rama analizada hasta el estado hoja (4,4,..., 4) que satisface cualquier que sea ésta. El inconveniente es que el número de pasos en la ruta encontrada puede ser mayor que el mínimo necesario.

En general [Russell, 96], cuando existen muchas soluciones al problema, la búsqueda por profundidad es preferible a la búsqueda por anchura, puesto que tiene más posibilidades de encontrar una solución explorando sólo una pequeña parte del árbol. En nuestro caso, se garantiza que basta con explorar una rama del árbol, cualquiera, para encontrar una solución.

Sin embargo, la secuencia de visitas encontrada de primeras mediante la búsqueda en profundidad, no tiene porqué ser óptima en cuanto a las preferencias del usuario, ni siquiera en cuanto a la longitud de la ruta. Es muy probable que existan otras ramas que partiendo del mismo estado inicial coloquen al usuario en un estado meta, con un número de visitas menor que el requerido en la primera rama evaluada.

En el ejemplo de la figura 8, la ruta encontrada evaluando la rama más a la izquierda en el árbol se compone de tres visitas, r = {visit(I2), visit(I3) y visit(I4)}.

Sin embargo, evaluando una rama situada más a la derecha es posible encontrar una ruta que tras una única visita a I4 conduce al usuario a un estado meta, s_m= (4,0,0,**4**,0,1,**2**).

En el ejemplo anterior, y siempre que exista una solución corta, la búsqueda en anchura produce mejores resultados que la búsqueda en profundidad. De modo que cuando la meta es poco ambiciosa y/o el conocimiento del usuario es muy alto, cabe esperar que la ruta tenga pocos pasos, y es sensato optar por realizar una búsqueda en anchura.

Por el contrario, si se desconoce la longitud estimada de la solución es menos arriesgado realizar una búsqueda en profundidad con la garantía de encontrar una solución en cada rama evaluada. Una **mejora** es que *la búsqueda no se detenga tras encontrar la primera solución, sino que se sigan buscando rutas, para poder elegir de entre todas las posibles, aquella con menor costo*. Para ello, una vez encontrado un nodo meta en un nivel *d*, se almacena la ruta y se vuelve al nivel anterior *d-1*, para seguir buscando.

En nuestro ejemplo esta vuelta atrás da lugar a que se encuentren, en el orden que se indican, tres rutas más: r'= {visit(I2), visit(I4)}, r''= {visit(I3), visit(I4)} y r'''= {visit(I4)}.

La **búsqueda en profundidad con vuelva atrás** requiere un espacio de memoria más modesto que la búsqueda en amplitud. Sólo es necesario guardar la ruta que va desde el nodo inicial hasta el nodo actual, junto con los nodos restantes no expandidos por cada nodo de la ruta. Siendo, la memoria necesaria para encontrar todas las posibles rutas es en el peor caso de O($n \times 4n$) [Russell, 96], donde n es el factor de ramificación y $4n$ la máxima profundidad para un nodo meta. La complejidad temporal vuelve a ser, sin embargo, igual a la obtenida para la búsqueda en amplitud, esto es, desde O(n^{4n}) para el peor caso hasta O(1) para el mejor.

A partir de las dos alternativas de búsqueda en profundidad planteadas: devolver la primera solución encontrada, o buscar todas las soluciones posibles para encontrar la óptima, optamos por una **propuesta mixta** que pretende llegar a un equilibrio entre el tiempo empleado en la búsqueda y la calidad de la ruta proporcionada al usuario.

En ésta, es de nuevo el usuario quien establece el tiempo máximo que desea esperar por la ruta. Durante dicho periodo de tiempo el sistema busca rutas usando la estrategia de búsqueda en profundidad. Una vez agotado el tiempo o finalizada expresamente la búsqueda, el sistema evalúa el costo de cada una de las rutas encontradas, devolviendo como solución la de menor costo. De esta forma, se garantiza que siempre se obtiene una solución buena (quizá no óptima), empleando para ello un consumo de tiempo y recursos razonable.

4.3 Búsqueda por profundización iterativa

Esta estrategia de búsqueda **realiza de forma iterativa, hasta encontrar la solución, una búsqueda limitada en profundidad**. En la búsqueda limitada en profundidad se pone *un límite a la profundidad máxima de la ruta*, ignorando todo el espacio de estados de las profundidades inferiores. Así, esta estrategia de búsqueda prueba, de menor a mayor, todos los limites de profundidad posibles: primero profundidad 0, luego profundidad 1, luego profundidad 2,... hasta encontrar la solución.

A continuación se muestra el algoritmo abstracto seguido para realizar la búsqueda en profundidad, utilizando distintos límites para ésta hasta dar con la solución.

```
profundidad = 0;
solución = ∅;
Mientras (solución = ∅) hacer
    solución = Búsqueda-limitada-por-profundidad(s_o, M', profundidad);
    profundidad = profundidad + 1;
```

Algoritmo 2. Búsqueda en profundidad

Observe en la figura 9, el resultado de aplicar las cuatro primeras iteraciones de este algoritmo en un árbol de búsqueda binario.

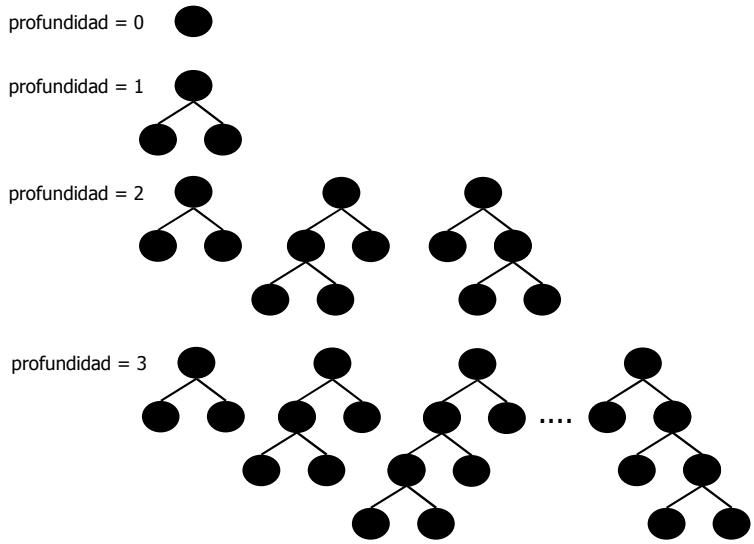


Figura 9: Búsqueda por profundización iterativa

Puede dar la impresión de que la búsqueda por profundización iterativa es un desperdicio, debido a la expansión de tantos estados tantas veces. Sin embargo, esta expansión múltiple resulta ser en realidad bastante pequeña, ya que los estados que se expanden muchas veces son los situados en los primeros niveles del árbol, y normalmente éstos van a tener pocos nodos. Esto ocurre por dos motivos:

a) La exponencialidad crece hacia abajo en el árbol.

b) Los primeros niveles suelen contener nodos que representan estados de conocimiento no muy altos, por lo que permiten el acceso a pocos ítems y generan una expansión de nodos reducida.

Concretamente si la solución se encuentra en una profundidad *d*, los nodos situados a profundidad *d-1* se han expandido una sola vez, los situados a profundidad *d-2* se expanden 2 veces,..., así hasta llegar al nodo raíz que se expande *d* veces. Siendo la cantidad total de expansiones en el peor caso, esto es suponiendo un factor de ramificación *n*, de: $d \times 1 + (d\text{-}1) \times n + (d\text{-}2) \times n^2 + ... + 2 \times n^{d-2} + 1 \times n^{d-1}$.

En este tipo de búsqueda se combinan las ventajas de las búsquedas preferente por profundidad y preferente por amplitud. Es *óptima y completa*, como la búsqueda por amplitud, al tiempo que la memoria que necesita es sólo la de la búsqueda preferente por profundidad. Sin embargo, puesto que en el peor caso, la complejidad temporal sigue siendo exponencial, debe permitirse al usuario decidir el tiempo que puede o quiere esperar por la ruta, pudiendo interrumpir la búsqueda en el momento que desee.

4.4 Búsqueda bidireccional

La búsqueda bidireccional es, básicamente, una búsqueda simultánea que **avanza a partir de un estado inicial y retrocede a partir de la meta**, y que se detiene cuando ambas búsquedas se encuentran en algún punto intermedio (véase la figura 10).

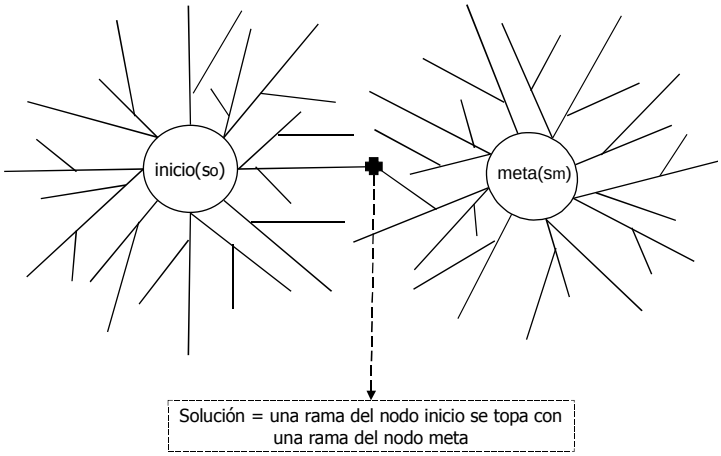


Figura 10: Búsqueda bidireccional

Esta estrategia puede reducir la complejidad temporal y espacial concretamente en $O(b^{d/2})$ [Russell, 96], siendo *b* el factor de ramificación y *d* la profundidad de la solución. Sin embargo, su aplicación al problema que nos ocupa, trae consigo una serie de problemas que no hacen factible su puesta en marcha.

Por un lado, lo normal es que *existan varios estados capaces de superar la prueba de meta*, por lo que la búsqueda hacia atrás, debe ser realizada de forma simultanea desde varios nodos del árbol. Para ello, es necesario identificar previamente el conjunto explícito de nodos meta, realizando, después, una **búsqueda de estado múltiple hacía atrás**, que obliga a calcular el predecesor de cada nodo implicado.

El problema es que se puede establecer el conjunto de estados que satisfacen una meta M', pero salvo para el estado de conocimiento "total", no se puede conocer cuáles de éstos forman parte del espacio de estados del problema. Con lo cual, siempre habría que partir hacia atrás del estado (4, 4,..., 4), aunque posiblemente existieran otros estados meta más próximos al estado inicial.

Por otro lado, para ir hacia atrás a partir de la meta, es necesario definir una función, **P**, capaz de **calcular los predecesores de un nodo** n_k, como todos aquellos para los que n_k es un sucesor. En nuestro problema, un nodo n_x es un predecesor de n_k, si aplicando sobre el estado representado en n_x el operador válido visit(i_j), se genera el estado de conocimiento representado en n_k (ecuación 14).

$$s_x \in P(s_k) \text{ si se cumple que: } s_k \in S(s_x)$$

$$\text{, esto es: } \exists i_j \text{ accesible en } s_x \text{ y } s_k = Ru(i_j)[s_x] \qquad (13)$$

Para obtener el conjunto de estados predecesores de s_k, esto es $P(s_k)$, hay que deshacer sobre dicho estado la aplicación del operador visit(i_j). Y una vez obtenido el estado resultante, s_x, comprobar si realmente es posible ejecutar sobre éste el operador visit(i_j), es decir si en s_x el ítem i_j es accesible. Sólo en caso afirmativo, el estado s_x es un verdadero predecesor de s_k. Obviamente, como de antemano no se conocen los operadores que pueden dar lugar al estado s_k, es necesario realizar esta maniobra con cada uno de los operadores posibles ($\forall i_j$).

El problema es que deshacer el operador visit(i_j) significa *deshacer la ejecución de la regla Ru(i_j)*. Y, mientras que un incremento sobre el grado de conocimiento de un ítem puede deshacerse sin demasiada complicación, una actualización fija, no ofrece ninguna información sobre el grado de conocimiento anterior del ítem, con lo que es imposible invertirla.

Por ejemplo, si dado el estado de conocimiento (4, 0, 3, 4, 3, 2, 4) intentamos deshacer la regla de actualización Ru(I4):

Si visitado(I4) entonces Fix-abs(I4, "total"), Inc-abs(I6, 1), Inc-abs(I7, 2);

, obtenemos el siguiente estado: (4, 0, 3, **?**, 3, **1**, **2**), donde hemos mantenido el conocimiento acerca de los ítems no actualizados en Ru(I4), hemos disminuido el conocimiento de I6 en un grado y el de I7 en dos grados, pero no podemos conocer el grado de conocimiento sobre I4 antes de aplicar la regla.

4.5 Búsqueda de coste uniforme

La búsqueda de coste uniforme [Nilsson, 87] modifica la estrategia preferente por amplitud[1] en el sentido de **expandir siempre el nodo de menor costo, medido por la función *g***, en lugar del nodo de menor profundidad. De esta manera, no se obtiene la ruta más corta, sino la ruta más apropiada considerando todas las preferencias del usuario: no sólo en cuanto a la longitud de la ruta sino también en cuanto a las características de los ítems que se visitan en ella (véase la función de costo en la sección 2.9).

Mediante la búsqueda por costo uniforme se puede encontrar la solución más barata siempre y cuando se satisfaga un requisito muy sencillo: el costo de la ruta nunca debe ir disminuyendo conforme avanzamos en ésta [Russell, 96].

La anterior restricción tiene sentido si se considera que el costo de la ruta en un nodo es la suma de los costos de los operadores que configuran la ruta hasta él. Entonces, basta con que todos los costos de los operadores sean no negativos para asegurar que el costo de una ruta nunca disminuye conforme se avanza en ella.

En nuestro problema, la función de costo *g* satisface el requisito especificado, de manera que se garantiza que la búsqueda de costo uniforme encuentra la ruta más barata sin tener que explorar el árbol en su totalidad. Para poder afirmar que se cumple el requisito, nos basamos en que:

- El costo g(i_j) asociado a un operador visit(i_j) siempre es mayor que cero (sección 2.9.2).

- El costo de la ruta es la suma del costo de los operadores que componen la ruta (sección 2.9.1).

---

[1] La búsqueda por amplitud se puede considerar una búsqueda de coste uniforme con g(n_x) = profundidad(n_x).

Por lo tanto, como se muestra en la ecuación 15, dada una ruta *r* compuesta de *n* visitas, visit(i_1), visit(i_2),..., visit(i_n), y costo g(r), al añadir una visita más a la ruta, visit(i_{n+1}), el nuevo costo, g'(r), va a ser forzosamente mayor.

$$g(r)= \sum_{j=1}^{n} g(i_j) < g'(r)= \sum_{j=1}^{n} g(i_j) + g(i_{n+1}) \qquad (14)$$

Siguiendo esta estrategia, en el árbol de la figura 11, primero se optaría por expandir el estado s_a ya que su costo asociado es menor que el de todos sus hermanos. Al expandirlo se obtiene el nodo s_g que es meta, detectado la ruta r1 = {visit(i_1), visit(i_4)}.

Sin embargo, puesto que el costo acumulado de s_a más el de s_g (0,3 + 0,52 = 0,82) supera el costo asociado a s_b (0,7), se vuelve hacia atrás para expandir s_b en busca de una posible solución más barata. Si s_b satisface la prueba de meta, la solución es r2 = {visit(i_2)}ya que el costo de esta ruta es menor que el de la ruta encontrada por s_a y que el costo del estado s_c que queda sin expandir.

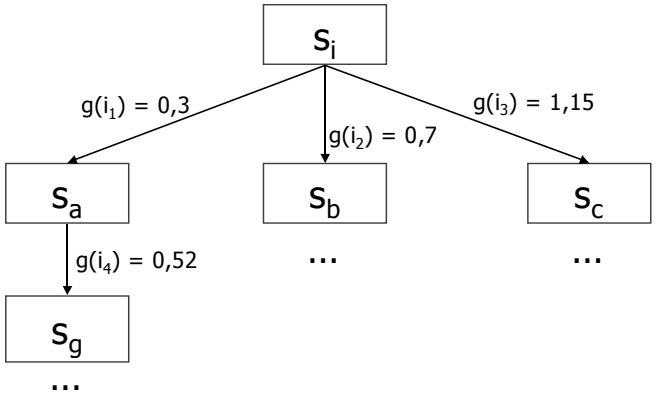


Figura 11: Búsqueda por coste uniforme

La búsqueda por costo uniforme es por tanto, *completa y óptima*. El problema es la complejidad temporal y espacial, que presenta el mismo orden que en la búsqueda preferente por amplitud, de modo que en el peor caso puede ser exponencial.

De nuevo cabe esperar que el tiempo efectivo para generar la ruta no sea excesivo y se permite al usuario detener la búsqueda a voluntad. Además, como alternativa se propone la **aplicación de la búsqueda sin vuelta atrás**. Esto es, en cada paso se elige el nodo con menor costo individual, hasta encontrar una solución.

En este caso, el número de nodos expandidos es 4×*n* en el peor caso y previsiblemente menor que *n* en el caso medio. Sin embargo, la ruta puede no ser óptima, ya que elegir el mejor nodo en cada paso no siempre es la mejor opción. A veces es preferible elegir un ítem con un costo un poco mayor pero que conduce a un estado a partir del cual el costo de ruta hasta la solución es muy bajo. En cualquier caso, aunque no sea óptima, la ruta encontrada va a contener ítems que se ajustan bastante a los gustos del usuario, ya que, precisamente, ese es el criterio utilizado en la búsqueda.

4.6 Búsqueda preferente por lo mejor: Búsqueda avara

Las anteriores estrategias de búsqueda no cuentan con información acerca de lo deseable o indeseable que resultará la expansión de un nodo, por lo cual son consideradas búsquedas ciegas. La búsqueda preferente por lo mejor es un tipo de **búsqueda respaldada con información**. En ésta, es necesario definir una función de evaluación que aproveche el conocimiento específico sobre el problema para decidir qué nodo conviene expandir primero.

La **función de evaluación** sobre un nodo produce un valor que sirve para representar lo deseable (o indeseable) que resulta su expansión. Los nodos se ordenan de forma que se expande primero aquél con mejor evaluación. Esta estrategia permite mejorar la eficiencia de los algoritmos de búsqueda ciega. Sin embargo, el nombre de búsqueda preferente por lo mejor es aceptable, pero no preciso [Russell, 96]. Ya que si de verdad fuese posible expandir desde el principio el mejor nodo, no se trataría de una búsqueda sino de un camino directo a la meta. Lo que sucede es que se escoge el nodo que parece ser mejor según la función de evaluación, aunque luego éste en realidad no resulte serlo.

La **búsqueda avara** es una estrategia de búsqueda preferente por lo mejor que consiste en reducir al mínimo el costo estimado para lograr la meta. Es decir, el nodo cuyo estado se considere más cercano al estado de la meta es el que se expande en primer lugar. En nuestro caso es posible estimar el costo que implica llegar a una meta desde un estado determinado, pero de una forma imprecisa. La estimación de costo es realizada por una **función heurística**, que representamos mediante la letra ***h***. Así, $h(n_x)$ siendo n_x un nodo del árbol de búsqueda, es *el costo estimado de la ruta más barata que une el estado s_x representado en el nodo con un estado meta s_m.*

La tarea de encontrar una función heurística no es fácil. Sin embargo, es frecuente que la solución de un **problema relajado**, esto es, con menos restricciones, constituya una buena heurística para el problema original. Desde un punto de vista formal, *h* puede ser cualquier función, cumpliendo un único requisito que $h(n_m) = 0$ cuando n_m representa un estado meta.

A partir de las anteriores premisas, planteamos dos heurísticas, h1 y h2, mostradas en las tablas 2 y 3 respectivamente. Ambas funciones devuelven coste 0 cuando se aplican sobre un estado meta y corresponden a distintas "relajaciones" del problema que pretendemos resolver.

Tabla 2: Una heurística para nuestro problema – h1

HEURÍSTICA 1
Es posible obtener una buena heurística, si relajamos el problema que nos ocupa en tres aspectos: a) Ignoramos las restricciones de accesibilidad de las reglas de conocimiento, esto es, suponemos que todos los ítems son accesibles en cualquier estado. b) Suponemos que en la regla de actualización de un ítem únicamente se actualiza el ítem visitado, c) y que la actualización sobre dicho ítem es "fija" y "absoluta" con un grado de conocimiento "total". Esto es, $\forall i_j$ Ru(i_j): Si visitado(i_j) entonces Fix-abs(i_j, "total");.

En el problema relajado, la secuencia de visitas, **r1**, necesaria en un estado $s_i = (K(i_1), K(i_2),..., K(i_n))$ para alcanzar un estado meta $s_m = (...,\_,v_k,\_,... ,\_, v_t,\_, ...)$ consta de una única visita a cada ítem i_j cuyo valor en la meta es mayor que el contemplado en s_i. Esto es, si se cumple que $K(i_k) < v_k$ entonces visit(i_k) $\in$ r1. Por supuesto, $K(i_k)$ siempre se considera mayor que _.

El costo estimado para llegar de s_i a s_m, **h1(s_i)**, es la suma del costo de cada uno de los ítems que deben visitarse en r1. Esta suma es calculada por la función de costo *g* aplicada sobre la secuencia de visitas r1, esto es **g(r1)**.

$s_i = (K(i_1), K(i_2), ..., K(i_n))$. M' = $\{(i_k, et_{SEM}^{k}),..., (i_t, et_{SEM}^{t})\}$

Paso 1. Inicializar la ruta. r1 = [].

Paso 2. Calcular la secuencia de visitas para llegar a la meta. r1:$s_i \rightarrow s_m$

Para cada ítem i_j tal que $K(i_j) \in s_i$ *hacer*

si $(i_j, et_{SEM}^{j}) \in$ M' y $K(i_j) <$ valor-numérico(et_{SEM}^{j}) *entonces* r1= r1 + {visit(i_j)}

Paso 3. Obtener el costo de la ruta. h1(s_i) = g(r1).

Tabla 3: Una heurística para nuestro problema – h2

HEURÍSTICA 2
Es posible encontrar otra heurística a partir de una especificación menos relajada del problema, en la cual se relajan sólo dos de las tres restricciones anteriores: a) Se ignoran las reglas de conocimiento, suponiendo que todos los ítems son accesibles en todos los estados de conocimiento posibles. b) Se supone que siempre la actualización de un ítem consiste en fijar su grado de conocimiento a "total". Esto es, $\forall$Ru(i_j), $\forall i_k$ Actualización(i_k) = Fix-abs(i_k, "total").
En este caso, el número de visitas necesarias para, a partir de un estado s_i, alcanzar un estado meta s_m no siempre va a coincidir con el número de ítems cuyo grado de conocimiento asociado es menor en s_i que en s_m. Esto se debe a que si existen dos submetas no satisfechas en s_i, la ejecución de una única regla de actualización sobre dicho estado puede satisfacer ambas si actualiza en su cuerpo a los dos ítems implicados. Concretamente, el número de visitas de la ruta **r2** que partiendo de s_i llega a s_m, coincide con el mínimo número de reglas de actualización necesarias para actualizar a todos los ítems que no satisfacen su meta de conocimiento en s_i. La ruta r2 se constituye con una visita al ítem cabeza de cada una de esas reglas. Y, la función heurística **h2(s_i)** devuelve, nuevamente, el costo de la ruta r2.
$s_i = (K(i_1), K(i_2),..., K(i_n))$. M' = $\{(i_k, et_{SEM}^{k}),..., (i_t, et_{SEM}^{t})\}$ **Paso 1**. Inicializar la ruta. r2 = []. **Paso 2**. Obtener un conjunto, **cj_m**, con todos los ítems que no satisfacen su submeta en s_i. $cj_m = \varnothing$. *Para cada* ítem i_j tal que $K(i_j) \in s_i$ *hacer* *si* $(i_j, et_{SEM}^{j}) \in$ M' y $K(i_j) <$ valor-numérico(et_{SEM}^{j}) *entonces* $cj_m = cj_m \cup \{i_j\}$ **Paso 3**. Construir la ruta que partiendo desde s_i llega hasta s_m. r2: $s_i \rightarrow s_m$ *Mientras* $cj_m \neq \varnothing$ *hacer* Buscar en el conjunto de reglas de actualización la regla Ru(i_j) que actualiza en su cuerpo el mayor número de ítems contenidos en cj_m.

Añadir a la ruta el ítem al que se asocia la regla, esto es, r2 = r2 + {visit(i_j)}

Para cada ítem $i_k \in cj_m$, *si* Actualización(i_k) $\in$ Ru(i_j) *entonces* $cj_m = cj_m - i_k$

Paso 4. Devolver el costo de la ruta encontrada. $h2(s_i) = g(r2)$.

Partiendo del ejemplo de las figuras 1 y 2, supongamos que en el estado s_o = (0,0,..., 0) hay que decidirse entre el estado s_1 = (4,0,0,0,0,0,0,0) obtenido tras visitar I1, el estado s_2 = (0,4,0,0,0,0,0,0) obtenido tras visitar I2 y el estado s_4 = (0,0,0,4,0,1,2) obtenido con la visita de I4.

Si la meta es M'={(I1, "total"), (I4, "total"), (I6, "medium"), (I7, total")}, un estado meta es cualquiera que se acople en la plantilla s_m = (4,_,_,4,_,2,4). El coste estimado para cada estado coincide con el coste de la ruta obtenida a partir de las condiciones de la heurística utilizada. En la tabla 4 se muestra la ruta desde cada estado hasta s_m según las heurísticas h1 y h2:

Tabla 4: Rutas estimadas por la heurística

	s1	**s2**	**s4**
h1	r1={I4,I6, I7}	r1={I1, I4, I6, I7}	r1={I1, I6, I7}
h2	r2 = {I4}	r2 = {I1, I4}	r2 = {I1, I4}

En este caso la heurística h1 aconseja expandir el estado s1 o s4, según sea el coste de I4 menor que el de I1 o viceversa. La heurística h2 se decide por la expansión de s1.

Si la meta es M'={(I4, "total"), (I6, "medium"), (I7, total")}, el estado meta es cualquiera que se acople a la plantilla s_m = (_,_,_,4,_,2,4). Y la ruta hasta s_m es para cada estado el que se indica en la tabla 5:

Tabla 5: Rutas estimadas por la heurística

	s1	**s2**	**s4**
h1	r1={I4,I6, I7}	r1={I4, I6, I7}	r1={I6, I7}
h2	r2 = {I4}	r2 = {I4}	r2 = {I4}

En este caso, la heurística h1 propone la expansión de s4, mientras que la heurística h2 considera las tres expansiones igual de favorables.

Observe que las dos heurísticas son optimistas, aún más h2. Ambas consideran siempre una actualización de conocimiento máxima y suponen que, en todo caso, es posible visitar el ítem que se necesita. De este modo, pueden aconsejar la elección de un nodo, suponiendo una ruta desde éste a la meta que realmente no puede seguirse tal cual, bien porque un ítem no es accesible cuando debe ser visitado o porque la ejecución de su regla de actualización no produce el conocimiento máximo que se espera.

Por ejemplo, para llegar desde s_1 = (4,0,0,0,0,0,0,0) hasta s_m = (4,_,_,4,_,2,4) se ha supuesto en la heurística h2 que basta con una visita a I4, ya que en su regla se actualizan I4, I6 e I7. Sin embargo, aplicando Ru(I4), nos damos cuenta de que realmente se requieren dos visitas, ya que tras la primera se llega a un estado (4,0,0,4,0,1,2) donde el incremento sobre I6 e I7 aún no alcanza el valor deseado.

En cualquier caso, se trata de una estimación que a pesar de no ser precisa permite ayudar a encontrar más rápido una buena solución.

4.7 Búsqueda A*

La **búsqueda avara** permite reducir al mínimo el costo de la meta, $h(n_x)$, con lo que se reduce de forma considerable el costo de la búsqueda. Desgraciadamente, este tipo de búsqueda no es óptima, aunque sí completa en el problema que nos ocupa. Por otra parte, la **búsqueda por costo uniforme**, reduce al mínimo el costo de la ruta, $g(n_x)$; es óptima y completa, pero puede ser muy ineficiente.

En el algoritmo A* [Nilsson, 87][Nilsson, 00] se combinan estas dos funciones mediante una suma. De esta forma la función de evaluación $\boldsymbol{f(n_x) = g(n_x) + h(n_x)}$ combina las ventajas de las dos estrategias. Dado que con $g(n_x)$ se calcula el costo de la ruta que va del nodo de partida al nodo n_x, y $h(n_x)$ es el costo estimado de la ruta más barata que va de n_x a la meta, tenemos que *$f(n_x)$ es el costo estimado de la solución más barata pasando por n_x*.

Por lo tanto, si se trata de encontrar la solución que sea más barata, es razonable probar primero con el nodo cuyo valor de *f* sea más bajo.

En la figura 12 se muestra la búsqueda A* con la heurística h2 que se ha realizado para satisfacer una meta M' = {(I3, "high"), (I5, "medium")}, teniendo el usuario un grado de conocimiento "nulo" sobre los ítems I3, I4, I5, I6 e I7 y "total" sobre I1 e I2.

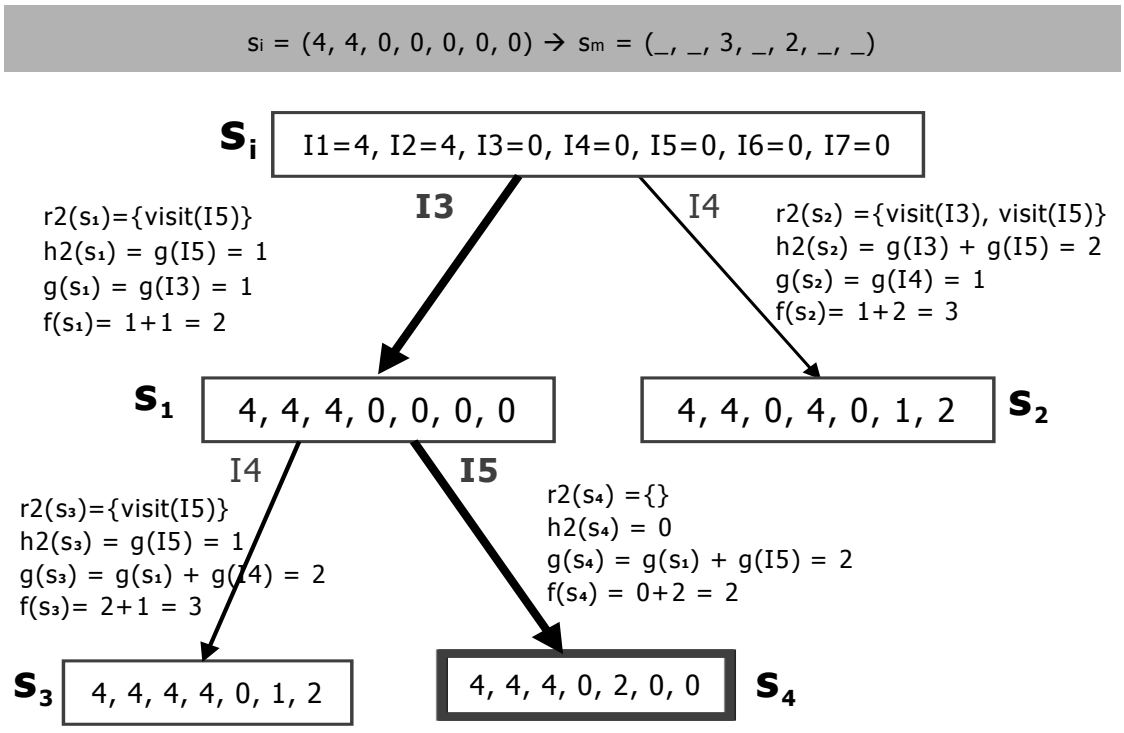


Figura 12: Búsqueda A*

Las reglas de actualización y conocimiento consideradas durante la búsqueda son las que se exponen en el ejemplo de la figura 1. Además, para el ejemplo se supone un peso de ruta corta, w(ruta-corta), igual a 1. Por lo tanto, el costo de una ruta coincide con el

número de visitas que ésta implica, ya que el costo de un ítem es siempre 1 independientemente de las características de éste. Esto es, $\forall i_j$, $g(i_j) = 1$.

Nótese en la figura, que la búsqueda A* encuentra sin necesidad de vuelta atrás la ruta óptima r2 = {visit(I3), visit(I5)} tan sólo generando cuatro estados diferentes: s_1, s_2, s_3 y s_4. Observe en cada estado, la ruta r2 estimada mediante la heurística h2 para llegar a la meta, y como la función de evaluación f suma el costo de r2 al costo de la ruta recorrida previamente para llegar a dicho estado. Esto es, $\forall s_j$, $f(s_j) = g(s_j) + h2(s_j)$.

Lo interesante de la búsqueda A* es que se demuestra que es *completa y óptima*, siempre que la función *h* sea una **heurística admisible**, esto es, satisfaga la siguiente restricción: *"la estimación realizada en h nunca debe sobreestimar el costo real que implica alcanzar la meta"*.

Estudiemos pues, la **admisibilidad de h2**: Cuando para cualquier ítem i_j, el costo de visita es 1, esto es $\forall i_j$ $g(i_j) = 1$, la heurística h2 satisface la restricción de admisibilidad, ya que el número de ítems que se visitan en la ruta estimada, r2, siempre es inferior o igual al número de visitas que se requieren en la ruta real hasta la meta.

Esto se debe a que h2 considera en cada paso la ruta más corta posible, al suponer que:

a) Son accesibles todos los ítems cuya regla de actualización conviene ejecutar.

b) Basta una única actualización para obtener conocimiento "total" sobre cada ítem actualizado, y por lo tanto, para satisfacer cualquier submeta impuesta sobre éste. Esto es, siempre se asume la actualización Fix-abs(i_k, "total").

Sin embargo, en la solución real, el número de visitas se va a ver incrementado por los siguientes motivos:

a) Debido a las reglas de conocimiento existentes, no es accesible el ítem cuya visita permite una mayor aproximación a la meta, siendo necesario visitar otros ítems previamente a éste.

b) Se requieren varias actualizaciones para alcanzar el conocimiento deseado sobre un ítem, por lo que es necesario ejecutar varias veces la regla de actualización, o lo que es lo mismo, visitar en más de una ocasión el ítem para el que se define. Por ejemplo, si en Ru(i_j) la actualización sobre i_k es Inc-abs(i_k, 1), $K(i_k) = 0$ y se desea $K(i_k) \geq 2$, se requieren al menos dos visitas sobre i_j.

En consecuencia, cuando el peso de ruta corta es 1, w(ruta-corta) = 1, la heurística h2 es admisible y la búsqueda A* completa y óptima. No obstante, cuando w(ruta-corta) es menor que 1, el costo de visitar un ítem depende del grado en que las características de éste se ajustan a los gustos del usuario. En consecuencia, puede existir una ruta más larga que la ruta r2 estimada en h2 que, sin embargo, presente un costo inferior al de ésta.

Esto ocurre siempre que el menor costo de los ítems implicados en la ruta compensa el mayor número de éstos, de manera que el costo total de la ruta queda por debajo de la estimación realizada en h2.

Por ejemplo, en la figura 12, para el estado s_1 se ha estimado que la ruta más corta hasta la meta es r2(s_1) = {visit(I5)}. Aunque, ciertamente es la ruta más corta, esto no quiere decir que sea la de menor costo.

Imaginemos, aunque no sea el caso, que existe otra ruta desde s_1, r(s_1) = {visit(I4), visit(I6)}, que también conduce a la meta. La longitud de la ruta *r* es obviamente mayor que la de *r2*. Pero si los ítems que contiene, I4 e I6, se acercan más que I5 a los gustos del usuario, quizá esto compense.

Por ejemplo, dados los siguientes costos: g(I5) = 0,9, g(I4) = 0,3 y g(I6) = 0,35, el costo de la ruta estimada es g(r2) = 0,9, mientras que el de la otra ruta es g(r) = 0,65, de modo que h2 no es admisible ya que sobreestima el costo real de alcanzar la meta.

Una forma de conseguir que h2 sea siempre admisible, consiste en modificar el paso 4 de la heurística (tabla 3) para que una vez obtenida la ruta más corta hasta la meta, r2, se calcule su costo considerando para cada ítem el costo mínimo posible. Este costo mínimo se denomina $\mathbf{g_{min}}$ y es 1 cuando w(ruta-corta) = 1. En otro caso se establece como el menor costo de un ítem incluido en la estructura de navegación actual. En la ecuación 16 se muestra el cálculo de g_{min} y la redefinición del paso 4.

$$\text{long} = \text{longitud}(r2) \qquad (15)$$

$$g_{min} = \mathit{mínimo}_{i_j}\left(g(i_j)\right) \text{ con } i_j \in I(EC_P^k)$$

$$h2(s_i) = \text{long} \times g_{min}$$

Con la modificación propuesta, la heurística h2 es admisible, y por lo tanto la búsqueda A* es óptima y completa en cualquier caso.

5. ALGORITMO GREEDY

Una alternativa a las estrategias de búsqueda explicadas anteriormente consiste en la aplicación de un algoritmo Greedy [Brassard, 96] [Cornen, 97]. Estos algoritmos conocidos también con el nombre de **avaros, glotones u oportunistas**, optan siempre por la mejor opción a corto plazo, sin reconsiderar posteriormente las opciones no elegidas. Esto les conduce de forma directa a una solución, que en muchos casos coincide con la óptima, pero que no tiene por qué serlo.

En el problema que nos ocupa, no siempre se puede llegar a una solución óptima a través de elecciones localmente óptimas. Por lo tanto, el algoritmo Greedy no garantiza una ruta óptima, pero sí una ruta satisfactoria. La principal ventaja es que el tiempo y la cantidad de memoria necesarias para encontrarla se reduce drásticamente al no realizarse vueltas atrás. Usando un algoritmo Greedy podemos garantizar que el sistema siempre es capaz de encontrar una ruta en un tiempo reducido, concretamente el orden es O(4×*n*) para el peor caso y previsiblemente menor que O(*n*) para el caso medio.

5.1 Elementos del algoritmo

El algoritmo Greedy que proponemos para resolver el problema de generación de rutas guiadas se compone de los siguientes elementos:

- **Candidato** (ϕ_k): Un candidato es un elemento susceptible de formar parte de la solución. En nuestro problema, cada candidato es representado mediante una pareja $\phi_k = (s_k, i_k)$ compuesta por un estado de conocimiento, s_k, y el ítem visitado para generarlo, i_k.

- **Lista de candidatos** (**C**): El conjunto de candidatos $C = \{\phi_1, \phi_2,..., \phi_n\}$ se obtiene, en cada paso, mediante la expansión del candidato actual $\phi_a = (s_a, i_a)$. Esta expansión genera tantos candidatos en C como ítems son accesibles en el estado de conocimiento s_a. El estado s_k incluido en el candidato ϕ_k se obtiene ejecutando, sobre el estado s_a, la regla de actualización $Ru(i_k)$ asociada a la visita del ítem i_k. Esto es, $s_k = Ru(i_k)[s_a]$.

- **Candidatos ya escogidos** (**S**): Es el conjunto de decisiones ya tomadas. Cuando se elige un candidato se inserta en este conjunto.

- **Prueba de meta** (**f-solución**): Función que determina si el conjunto de candidatos seleccionado, S, es una solución al problema. En este caso, la comprobación sólo afecta al último candidato elegido y consiste en ver si el estado de conocimiento representado en éste satisface los todos requisitos de conocimiento de la meta M'.

  Dada la meta $M' = \{(i_s, et_{SEM}^{s}),..., (i_t, et_{SEM}^{t})\}$, el conjunto de candidatos $S = \{\phi_1, \phi_2,..., \phi_{k-1}, \phi_k\}$ y el candidato $\phi_k = (i_k, s_k)$ con $s_k = (K(i_1), ..., K(i_s), ..., K(i_t), ..., K(i_n))$.

  f-solución(S, M') = prueba-meta(s_k), por lo que f-solución(S, M')= *true* sólo si $K(i_s) \geq$ valor-numérico(et_{SEM}^{s}) $\wedge ... \wedge K(i_t) \geq$ valor-numérico(et_{SEM}^{t}).

- **Función objetivo** (**f-objetivo**): Está función indica lo adecuada que es la elección de un candidato. La función objetivo aplicada a un candidato ϕ_k, f-objetivo(ϕ_k), devuelve el costo de dicho candidato y se calcula sumando el costo de sus dos componentes: ítem (i_k) y estado (s_k). Cuanto menor sea su valor más adecuado es ese candidato. Por lo tanto, es un valor a minimizar.

La función de **costo de un ítem**, **g'(i_k)**, se redefine de tal forma que sólo tiene en cuenta la *idoneidad de las características del ítem i_k*, ignorándose por ahora, la longitud de la ruta. Puesto que el peso asignado a este conjunto de preferencias se obtiene como uno menos el peso asignado por el usuario a la preferencia de ruta-corta, debe ser precisamente éste, 1-w(ruta-corta), el costo máximo de un ítem. Obviamente, el costo de un ítem debe estar más próximo a cero en cuanto que sus características se acerquen más a los gustos del usuario (véase la ecuación 17).

$$g'(i_k) = \frac{1 - \sum_{at^i \in A} w(at^i) \times g(i_k, at^i)}{2} \times \left(1 - w(ruta - corta)\right) \qquad (16)$$

Observe que en la sumatoria $\sum w(at^i) \times g(i_k, at^i)$ se utiliza la distribución inicial de pesos sobre las características, esto es $w(at^i)$ en lugar de $w'(at^i)$. Esto significa que no se ha

descontado el peso de la preferencia por ruta-corta. De modo que, $\sum w(at^i) = 1$, y por lo tanto, el valor de la sumatoria va a ser 1 para el mejor ítem y -1 para el peor ítem.

Esto hace que el numerador del primer término de la ecuación 17 tenga valor 0 cuando las características del ítem son las que más gustan al usuario y 2 cuando son las que menos. Al dividir entre 2 trasladamos este valor al intervalo [0,1] de modo que 0 es el costo del mejor ítem y 1 el del peor. Finalmente multiplicamos este valor por el término 1-w(ruta-corta) para conseguir que en el peor caso el costo del ítem sea exactamente ese.

La función de **costo de un estado**, **$g'(s_k)$**, es la que se ocupa de calcular la *distancia que existe entre el estado actual, s_k, y la meta*. Por lo tanto, debe tener en cuenta la importancia establecida por el usuario acerca de la longitud de la ruta. La función $g'(s_k)$ devuelve un valor alto si el estado s_k dista mucho del estado objetivo establecido en la meta. De modo que, el costo más alto posible coincide, precisamente, con el grado de preferencia del usuario por las ruta cortas, esto es w(ruta-corta).

Un estado de conocimiento s_k cercano al objetivo obtiene un valor bajo en $g'(s_k)$, siendo cero en el mejor caso, esto es cuando se trata de un estado meta. Y, un estado s_k tiene costo w(ruta-corta), únicamente si, entre todos los estados candidatos, es el más alejado de la meta.

La distancia entre un estado $s_k = (K(i_1), K(i_2), ..., K(i_n))$ y un estado meta se denomina **$d(s_k, M')$**, y se calcula como la suma del mínimo número de grados de conocimiento en que debe incrementarse el conocimiento de cada ítem en s_k para que satisfaga la submeta impuesta sobre él en M'. En el algoritmo 3 puede verse el cálculo de esta distancia para cada uno de los candidatos.

```
Para cada ¢_k ∈ C hacer
    d(s_k, M') = 0.
    Para cada (i_j, et_SEM^j) ∈ M' hacer
        Si K(i_j) < valor-numérico(et_SEM^j) entonces
        d(s_k, M') = d(s_k, M') + (valor-numérico(et_SEM^j) - K(i_j))
```

Algoritmo 3. Distancia de cada candidato a la meta

Como variante, se puede establecer el costo de un estado s_k como el número de ítems que en dicho estado tienen un grado de conocimiento inferior al requerido en la meta, en lugar de cómo el número total de grados que separan el estado actual del estado meta. Para ello, basta reemplazar la última línea del algoritmo 3, $d(s_k, M') = d(s_k, M') + (\text{valor-numérico}(et_{SEM}^j) - K(i_j))$, por esta otra: $d(s_k, M') = d(s_k, M') + 1$.

En cualquier caso, una vez obtenida la distancia que existe del estado s_k a la meta, su costo, $g'(s_k)$, se obtiene multiplicando esta distancia, $d(s_k, M')$, por el resultado de dividir el peso de ruta-corta, w(ruta-corta), entre la distancia más larga de un candidato a la meta. El cálculo realizado se especifica en el algoritmo 4.

```
d_max(C,M') = máximo_C {d(s_1, M'), ..., d(s_n, M')}
Para cada ¢_k ∈ C hacer
```

$$g'(s_k) = d(s_k, M') \times \frac{w(ruta-corta)}{d_{\max}(C, M')}$$

Algoritmo 4. Costo de cada candidato

De esta forma se consigue que el costo del candidato (o candidatos) más alejado de la meta sea w(ruta-corta) y que dicho costo disminuya conforme más se acerque a ésta. Obteniéndose, obviamente, valor cero cuando se trata de un estado meta, s_m, para el cual la distancia d(s_m, M`) es cero.

El costo de un candidato es la suma del costo del ítem y el estado que lo forman. Por lo tanto, el resultado de la función objetivo para un candidato $¢_k$ se obtiene combinando g'(i_k) y g'(s_k) tal y como se muestra en la ecuación 18.

$$\text{f-objetivo}(¢_k) = g'(i_k) + g'(s_k) \qquad (17)$$

De este modo, el costo más alto de un candidato $¢_k$ es 1 cuando su estado es el más alejado de la meta, g'(s_k) = w(ruta-corta), y el ítem visitado presenta en todas las características el valor que más disgusta al usuario, g'(i_k) = 1- w(ruta-corta). Por el contrario, el costo más bajo para un candidato $¢_k$ es cero cuando se trata de un estado meta que además supone la visita a un ítem que satisface de lleno todos los gustos del usuario. Entre ambos casos extremos, el costo de un candidato depende de cuál sea su distancia a la meta (hasta un máximo de w(ruta-corta)) y como de apropiado es el ítem que se visita (hasta un máximo de 1-w(ruta-corta)).

El costo de un ítem puede ser precalculado, de forma que se recalcula únicamente cuando el usuario modifica sus preferencias al respecto o, en el caso, poco frecuente, de que el autor cambie el valor de alguna de sus características. El costo de un estado depende de la meta actual y por lo tanto debe ser calculado sobre la marcha.

- **Función de selección** (**f-selección**): Es la función que determina el candidato más prometedor. Para ello, siempre elige el candidato que minimiza la función objetivo.

  f-selección(C) = $¢_k$ tal que f-objetivo($¢_k$) = minimo_C{f-objetivo($¢_1$),..., f-objetivo($¢_n$)}

En nuestro problema no es necesaria ninguna función para determinar si un conjunto de candidatos es *completable*, ya que sabemos que siempre lo es. Como se ha argumentado anteriormente (sección 3.3.2), desde cualquier estado existe una secuencia de visitas que conducen a un estado meta. Por lo cual, siempre es posible encontrar una nueva solución añadiendo candidatos al conjunto actual. Sin embargo, es obvio que, añadir más candidatos a una solución no tiene sentido, ya que se aumentaría innecesariamente la longitud de la ruta. Por esto, el algoritmo acaba en cuanto que se satisface la prueba de meta. Esto es, cuando f-solución(S, M') devuelve el valor lógico de verdad.

5.2 Aplicación del algoritmo

A partir de los elementos especificados en la sección anterior, el esquema del algoritmo Greedy puede escribirse en pseudocódigo de la siguiente forma:

```
algoritmo Greedy (ENTRADA s_o: estado-inicial, M': meta; SALIDA r: ruta)

    variables
```

```
        C: lista de candidatos
        S: candidatos seleccionados
        ¢a: candidato actual
        sa: estado del candidato actual
        ia: ítem del candidato actual
    principio
        r = {},  S = ∅
        sa = so
        Mientras  (f-solución(S, M') ≠ true) hacer
            principio
                C = expandir(sa)
                ¢a =  f-selección(C)
                S = S ∪ {¢a}
                sa = estado(¢a)
            fin
        Para cada ¢a ∈ S  hacer
            principio
                ia  = ítem(¢a)
                r = r + {visit(ia)}
            fin
fin
```

Algoritmo 5. Greedy para encontrar una ruta guiada

Recapitulando, el algoritmo Greedy propuesto intenta sacar en cada paso el mayor provecho posible. Para ello, elige el candidato con menor costo, sin preocuparse de si a largo plazo esta elección es óptima o no. De modo que, una vez incorporado un candidato a la solución, permanece en ella hasta el final.

El costo de un candidato $¢_k$ depende de dos aspectos:

a) la cercanía del estado a la meta, $g'(s_k)$, y

b) las características del ítem visitado, $g'(i_k)$.

Por ello, un candidato $¢_1 = \{s_1, i_1\}$ tiene un costo menor que $¢_2 = \{s_2, i_2\}$ en cuanto que:

a) la diferencia de conocimiento que existe entre la meta y el estado s_1 es menor que la existente para s_2, y

b) las características del ítem i_1 se ajustan mejor que las características de i_2 a los gustos del usuario.

La importancia que tiene cada uno de estos dos aspectos se obtiene del propio modelo de usuario, **w(ruta-corta)** para **a)** y **1- w(ruta-corta)** para **b)**. Es decir, el costo de un estado, $g'(s_k)$, pertenece al intervalo cerrado [0, w(ruta-corta)] y el costo de un ítem, $g'(i_k)$, a [0, 1- w(ruta-corta)]. Puesto que ambos costos se suman para obtener el costo del candidato, f-objetivo($¢_k$), siempre está comprendido en el intervalo [0, 1].

A continuación se aplica el algoritmo Greedy sobre el ejemplo de las figuras 1 y 2, considerando un estado inicial s_o = (0,0,0,0,0,0,0) y una meta M' = {(I2, "high"), (I4, "medium"), (I7, "total")}.

Suponemos, también, que el usuario ha establecido un peso de ruta corta, w(ruta-corta), igual a 0,5. Por lo tanto, ambos aspectos, características del ítem y longitud de la ruta, tienen la misma importancia para él. Siendo 0,5 el costo más alto posible tanto para un ítem, $g'(i_j)$, como para un estado, $g'(s_k)$.

Suponemos que el costo asociado a cada ítem, $g'(i_j)$, ha sido previamente calculado en función de sus características, los gustos establecidos por el usuario para cada característica y el peso dado a ésta. Obteniéndose los costos que se muestran en la siguiente tabla:

Tabla 6: Costos de los ítems implicados en la búsqueda

g'(I1)	g'(I2)	g'(I3)	g'(I4)	g'(I5)	g'(I6)	g'(I7)
0,05	0,1	0,04	0,1	0,23	0,3	0,35

Inicialización

$S = \varnothing$

r = {}

s_a = (0,0,0,0,0,0,0)

Primera iteración

C = expandir(s_a) = { ((4,0,0,0,0,0,0), I1), ((0,4,0,0,0,0,0), I2) ,((0,0,0,4,0,1,2), I4) }

d(s1, M') = (_, 3, _, 2, _, _, 4) - (4,**0**,0,**0**,0,0,**0**) = 9

d(s2, M') = (_, 3, _, 2, _, _, 4) - (0,**4**,0,**0**,0,0,**0**) = 6

d(s4, M') = (_, 3, _, 2, _, _, 4) - (0,**0**,0,**4**,0,1,**2**) = 5

d_{max}(C, M') = max {d(s1, M'), d(s2, M'), d(s4, M')} = max {9, 6, 5} = 9

$g'(s1) = 9 \times (0,5/9) = 0,5$. f-objetivo(¢1) = g'(s1) + g'(I1) = 0,5 + 0,05 = 0,55

$g'(s2) = 6 \times (0,5/9) \approx 0,3$. f-objetivo(¢2) = g'(s2) + g'(I2) = 0,3 + 0,1 = 0,4

$g'(s4) = 5 \times (0,5/9) \approx 0,27$. f-objetivo(¢4) = g'(s4) + g'(I4) = 0,27 + 0,1 = 0,37

f-selección (C) = ¢4 = ((0,0,0,4,0,1,2), I4), $S = S \cup \{((0,0,0,4,0,1,2), I4)\}$

f-solución(S, M') ? False, entonces s_a = s4.

Segunda iteración

C = expandir(s_a) ={ (s1', I1), (s2', I2), (s4', I4), (s5, I5), (s6, I6), (s7, I7) }

d(s1', M') = (_, 3, _, 2, _, _, 4) - (4,**0**,0,**4**,0,1,**2**) = 5

d(s2', M') = (_, 3, _, 2, _, _, 4) - (0,**4**,0,**4**,0,1,**2**) = 2

d(s4', M') = (_, 3, _, 2, _, _, 4) - (0,**0**,0,**4**,0,2,**4**) = 3

d(s5, M') = (_, 3, _, 2, _, _, 4) - (2,**0**,0,**4**,2,1,**2**) = 5

d(s6, M') = (_, 3, _, 2, _, _, 4) - (0,**1**,0,**4**,0,4,**2**) = 4

d(s7, M') = (_, 3, _, 2, _, _, 4) - (0,**0**,0,**4**,0,1,**4**) = 3

d_{max}(C, M') = max {5, 2, 3, 5, 4, 3} = 5

g'(s1') = 5 × (0,5/5) = 0,5. f-objetivo(¢1) = 0,5 + 0,05 = 0,55

g'(s2') = 2 × (0,5/5) = 0,2. f-objetivo(¢2) = 0,2 + 0,1 = 0,3

g'(s4') = 3 × (0,5/5) = 0,3. f-objetivo(¢4) = 0,3 + 0,1 = 0,4

g'(s5) = 5 × (0,5/5) = 0,5. f-objetivo(¢5) = 0,5 + 0,23 = 0,73

g'(s6) = 4 × (0,5/5) = 0,4. f-objetivo(¢6) = 0,4 + 0,3 = 0,7

g'(s7) = 3 × (0,5/5) = 0,3 f-objetivo(¢7) = 0,3 + 0,35 = 0,65

f-selección(C) = ¢2 = ((0,4,0,4,0,1,2), I2), S = S ∪ {((0,4,0,4,0,1,2), I2)}

f-solución (S, M') ? False, entonces s_a = s2'

Tercera iteración

C = expandir(s_a) ={ (s1'', I1), (s2'', I2), (s3, I3), (s4'', I4), (s5', I5), (s6', I6), (s7', I7) }

d(s1'', M') = (_, 3, _, 2, _, _, 4) - (4,**4**,0,**4**,0,1,**2**) = 2

d(s2'', M') = (_, 3, _, 2, _, _, 4) - (0,**4**,0,**4**,0,1,**2**) = 2

d(s3, M') = (_, 3, _, 2, _, _, 4) - (0,**4**,4,**4**,0,1,**2**) = 2

d(s4'', M') = (_, 3, _, 2, _, _, 4) - (0,**4**,0,**4**,0,2,**4**) = 0

d(s5', M') = (_, 3, _, 2, _, _, 4) - (4,**4**,0,**4**,4,1,**2**) = 2

d(s6', M') = (_, 3, _, 2, _, _, 4) - (0,**4**,0,**4**,0,4,**2**) = 2

d(s7', M') = (_, 3, _, 2, _, _, 4) - (0,**4**,0,**4**,0,1,**4**) = 0

d_{max}(C, M') = max {2, 2, 2, 0, 2, 2, 0} = 2

g'(s1'') = g'(s2'') = g'(s3) = g'(s5') = g'(s6') = 2 × (0,5/2) = 0,5

g'(s4'') = g'(s7') = 0

f-objetivo(¢1) = 0,5 + 0,05 = 0,55

f-objetivo(¢2) = 0,5 + 0,1 = 0,6

f-objetivo(¢3) = 0,5 + 0,04 = 0,54

f-objetivo(¢4) = 0 + 0,1 = 0,1

f-objetivo(¢5) = 0,5 + 0,23 = 0,73

f-objetivo(¢6) = 0,5 + 0,3 = 0,8

f-objetivo(¢7) = 0 + 0,35 = 0,35

f-selección (C) = ¢4 = (0,4,0,4,0,2,4), I4), $S = S \cup \{(0,4,0,4,0,2,4), I4)\}$

f-solución(S, M') ? Sí, entonces devolver la ruta.

Solución r = {visit(I4), visit(I2), visit(I4)}

Observe, cómo en este caso el algoritmo Greedy ha encontrado una solución óptima en tan sólo tres iteraciones.

6. METAS SOBRE CONCEPTOS

A lo largo del presente capítulo se han propuesto y analizado distintos métodos que el sistema puede utilizar para generar una ruta guiada [Medina, 03]. La ruta es una secuencia ordenada de visitas, que partiendo del estado de conocimiento actual del usuario, le permita alcanzar un nuevo estado donde se satisfacen todas las submetas que el mismo ha impuesto sobre los ítems de la estructura de navegación actual.

En esta sección se plantean las particularidades que existen cuando una submeta, en lugar de estar definida sobre un ítem, lo está sobre un concepto. Estudiando, en consecuencia, las modificaciones necesarias sobre los procedimientos explicados en las secciones anteriores.

6.1 Ámbito de una submeta conceptual

En primer lugar, como se expuso en el capítulo 19 (Ítems deseables), es necesario determinar cuáles de las submetas conceptuales tienen validez en la estructura de navegación actual. Esto es, sólo se pueden considerar las submetas, $(c_i, et_{SEM}^{\ i}) \in M$, que el usuario puede satisfacer durante la navegación de la estructura elegida, $EC_A^{\ j} = (EC_M, Rw, MU, EC_P^{\ k}, RTnb^i, Ro^i, Ru^j, Rk^j)$.

Concretamente, para que un concepto meta c_i sea considerado en una estructura $EC_A^{\ j}$, éste debe mostrarse en la presentación sobre la que ésta se define, $c_i \in C(EC_P^{\ k})$. Pero además, es necesario que el grado de conocimiento impuesto sobre él, $et_{SEM}^{\ i}$, pueda ser obtenido visitando los ítems incluidos en la misma.

El grado de conocimiento máximo que el usuario puede alcanzar sobre un concepto c_i recorriendo la presentación $EC_P^{\ k}$, depende del estado de conocimiento del usuario antes de comenzar a navegarla, $estado_K$, los ítems asociados a c_i en dicha estructura, $<c_i, r_f, i_j> \in Af(EC_P^{\ k})$, y la regla de peso definida para el concepto, $Rw(c_i)$.

Este conocimiento máximo, $\mathbf{K_{max}^{\ k}(c_i)}$, se obtiene aplicando la regla de peso $Rw(c_i)$ sobre un estado donde el conocimiento de los ítems $c_i.i_j$ no incluidos en la $EC_P^{\ k}$ coincide con el que aparece registrado en $estado_K$, mientras que el conocimiento de los ítems $c_i.i_j$ sí incluidos en $EC_P^{\ k}$ se considera "total" (véase el capítulo 19).

De esta forma para que una submeta (c_i, et_{SEM}^i) se tenga en cuenta durante la generación de la ruta guiada, esto es, (c_i, et_{SEM}^i) ∈ M', debe cumplirse que:

$$(c_i, et_{SEM}^i) \in M \wedge c_i \in C(EC_P^k) \wedge K_{max}^k(c_i) \geq \text{valor-numérico}(et_{SEM}^i) \quad (18)$$

6.2 Redefinir la prueba de meta

Otra cuestión, es redefinir la prueba de meta, ya que ahora no basta con comprobar si se satisfacen las submetas de conocimiento impuestas en M' sobre los ítems de la estructura actual, sino que además es necesario evaluar las que afectan a los conceptos.

Para determinar si se satisface la submeta impuesta sobre un concepto c_i, es necesario conocer el grado de conocimiento actual sobre dicho concepto, **$K(c_i)$**. Sin embargo, esta información, a diferencia de para los ítems, no se encuentra almacenada explícitamente en los estados del espacio de búsqueda.

No obstante, no supone ningún problema obtener el conocimiento acerca de un concepto. Para ello, basta aplicar la regla de peso del concepto, $Rw(c_i)$, usando el grado de conocimiento almacenado para cada $c_i.i_j$ en el estado evaluado, y en su defecto, el que aparece registrado en el modelo de usuario.

Dada una meta M'y el estado de conocimiento $s_k = (K(i_1), K(i_2),..., K(i_n))$ la prueba de meta se redefine como sigue:

prueba-meta(s_k, M') = *true* si: (19)

$\forall(i_j, et_{SEM}^j) \in M'$, $K(i_j) \geq$ valor-numérico(et_{SEM}^j)

$\forall(c_i, et_{SEM}^i) \in M'$, $Rw(c_i)[s_k, estado_K] \geq$ valor-numérico(et_{SEM}^i)

, donde $Rw(c_i)[s_k, estado_K]$ indica la evaluación de la regla $Rw(c_i)$ de acuerdo a lo explicado, esto es:

$Rw(c_i): K(c_i) = \sum_{j=1}^{r} w_j \times K(c_i.i_j)$, con $K(i_j) \in s_k$ si $i_j \in I(EC_P^k)$ ó $K(i_j) \in estado_K$ si $i_j \notin I(EC_P^k)$.

6.3 Otras consideraciones

Una vez que se ha establecido cuándo se incluye en M' una submeta definida sobre un concepto (sección 6.1), y cómo se averigua si un estado es o no meta teniendo, también, en cuenta las submetas conceptuales impuestas en M' (sección 6.2). El resto del proceso realizado por el sistema para averiguar la ruta coincide con lo explicado en las secciones 1 hasta 5. Excepto, en dos aspectos, las heurísticas h1 y h2 propuestas en la búsquedas respaldada con información (avara y A*), y el costo del estado, $g'(s_k)$, calculado en el algoritmo Greedy.

6.3.1 Adaptación de $g'(s_k)$ en el algoritmo Greedy

En el caso del algoritmo Greedy propuesto, la adaptación es sencilla. La distancia de un estado $s_k = (K(i_1), K(i_2), ..., K(i_n))$ a la meta sigue siendo el número de grados de

conocimiento en que debe incrementarse el estado s_k para satisfacer M'. Esto implica, restar al valor mínimo exigido en cada submeta el valor que actualmente se tiene para el elemento implicado, ya sea un ítem o un concepto.

Si se trata de un ítem, i_j, se utiliza el conocimiento representado en el estado, $K(i_j)$, y si se trata de un concepto, c_i, se obtiene el conocimiento sobre éste a través de la ejecución de su regla de peso, $Rw(c_i)[s_k, estado_K]$ (véase la modificación del algoritmo 3).

Para cada $\varsigma_k \in C$ *hacer*

$d(s_k, M') = 0.$

Para cada $(i_j, et_{SEM}^j) \in M'$ *hacer*

Si $K(i_j) <$ valor-numérico(et_{SEM}^j) *entonces*

$d(s_k, M') = d(s_k, M') +$ (valor-numérico$(et_{SEM}^j) - K(i_j)$)

Para cada $(c_i, et_{SEM}^i) \in M'$ *hacer*

$K(c_i) = Rw(c_i)[s_k, estado_K]$

Si $K(c_i) <$ valor-numérico(et_{SEM}^i) *entonces*

$d(s_k, M') = d(s_k, M') +$ (valor-numérico$(et_{SEM}^j) - K(c_i)$)

Algoritmo 3'. Distancia de cada candidato a la meta

6.3.2 Adaptación de las heurísticas h1 y h2

En el caso de las heurísticas, la adaptación no es tan directa. En **h1** se estima que la ruta hasta la meta está formada por una visita a cada uno de los ítems para los que aún no se satisface su submeta. En el caso de un concepto, la submeta debe ser satisfecha visitando, en la estructura actual, uno o varios de los ítems asociados a éste. El problema es que pueden existir varias combinaciones de visitas distintas que consiguen satisfacer la misma submeta.

Para establecer el ítem o conjunto de ítems que debe visitar el usuario para alcanzar el grado de conocimiento, et_{SEM}^i, establecido en la submeta de un concepto c_i, se va a seguir un planteamiento optimista. De acuerdo a éste, se obtiene el conjunto más pequeño de ítems, que al ser visitados consiguen que el conocimiento del concepto c_i supere su submeta.

Denominamos a este conjunto **$v(c_i)$**, y obviamente, sólo puede contener ítems que aparezcan en la estructura de navegación actual, ya que sobre los otros es imposible aumentar el conocimiento. Los ítems se incluyen uno a uno en el conjunto $v(c_i)$. Un ítem i_j se incluye antes que otro i_k si puede aportar más ganancia de conocimiento sobre el concepto c_i al que ambos se asocian. Esta ganancia, **$gan(i_j)$**, depende del peso que tenga el ítem en la regla, w_j, y de cuántos grados de conocimiento le faltan en el estado actual para llegar a ser "total" (véase la ecuación 20).

$$Rw(c_i): K(c_i) = \sum_{j=1..r} w_j \times K(c_i.i_j)$$

$$s_k = (K(i_1), K(i_2), ..., K(i_n))$$

$$gan(i_j) = (4 - K(i_j)) \times w_j \qquad (20)$$

Después de añadir un nuevo ítem al conjunto $v(c_i)$ se reevalúa la regla de peso para comprobar si suponiendo conocimiento "total" sobre todos los ítems incluidos, el conocimiento calculado para el concepto c_i cumple su submeta, o sea $K(c_i) \geq$ valor-numérico(et_{SEM}^{i}). Cuando esto ocurre el conjunto está completo y contiene el mínimo número de ítems que hay que visitar para satisfacer la submeta del concepto (véase el algoritmo 6).

$(c_i, et_{SEM}^{i}) \in M'$ y $Rw(c_i)$: $K(c_i) = \sum_{j=1..r} w_j \times K(c_i.i_j)$

$v(c_i) = \varnothing$.

$s_k' = s_k$.

Mientras $Rw(c_i)[s_k', estado_K] <$ valor-numérico(et_{SEM}^{i}) *hacer*

Para cada i_u *tal que*

$i_u \in I(EC_P^k) \wedge i_u \notin v(c_i) \wedge w_u =$ máximo $_{ij \notin v(ci)}$ {gan(i_1), gan(i_2), ...}

Hacer

$v(c_i) = v(c_i) \cup \{i_u\}$

$s_k' = s_k'$ donde $K(i_u) =$ "total"

Algoritmo 6. Conjunto $v(c_i)$ para satisfacer (c_i, et_{SEM}^{i})

Supongamos que el concepto *Hatha-Yoga*, incluido en la estructura de navegación de la figura 1, tiene asociada la siguiente regla de peso:

Rw(Hatha-Yoga): K(Hatha-Yoga) = 0,4×K(I5) + 0,25×K(I6) + 0,2×K(I7) + 0,15×K(I8)

De los cuatro ítems implicados, sólo tres, I5, I6 e I7, están presentes en la estructura actual. Por lo tanto, para el ítem I8, el grado de conocimiento del usuario se va a mantener con el valor que tenía antes de comenzar su navegación. Supongamos por ejemplo K(I8) = **1** en $estado_K$.

Dado un estado s_k = (0, 3, 2, 4, **0**, **1**, **0**) y la submeta (Hatha-Yoga, "high"), tenemos que ésta no se cumple puesto que:

$$K(\text{Hatha-Yoga}) = 0{,}4 \times \mathbf{0} + 0{,}25 \times \mathbf{1} + 0{,}2 \times \mathbf{0} + 0{,}15 \times 1 = 0{,}4 \text{ ("null")} < \text{"high"}$$

Es necesario obtener el conjunto mínimo de ítems que hay que visitar para satisfacer la submeta. Para ello, se calcula la ganancia de cada ítem:

$gan(I5) = 4 \times 0{,}4 = 1{,}6.$

$gan(I6) = 3 \times 0{,}25 = 0{,}75.$

$gan(I7) = 4 \times 0{,}2 = 0{,}8.$

El ítem I5 se incluye primero en el conjunto v(Hatha-Yoga) porque proporciona la mayor ganancia. Ahora, se reevalúa la regla de peso considerando K(I5) = "total". Esto es, s_k' = (0, 3, 2, 4, **4**, **1**, **0**).

$$K(\text{Hatha-Yoga}) = 0{,}4 \times \mathbf{4} + 0{,}25 \times \mathbf{1} + 0{,}2 \times \mathbf{0} + 0{,}15 \times 1 = 2 \text{ ("medium")} < \text{"high"}$$

Puesto que sigue sin satisfacerse la submeta, se añade al conjunto el siguiente ítem con mayor ganancia, esto es I7. Ahora, tras reevaluar la regla con s_k' = (0, 3, 2, 4, **4**, **1**, **4**), sí se satisface la submeta.

$$K(\text{Hatha-Yoga}) = 0{,}4 \times 4 + 0{,}25 \times 1 + 0{,}2 \times \mathbf{4} + 0{,}15 \times 1 = 2{,}8 \text{ (“high”)} \geq \text{“high”}$$

El conjunto mínimo de ítems que hay que visitar finalmente es: v(Hatha-Yoga) = {I5, I7}.

Con la modificación propuesta, la ruta estimada con h1 para alcanzar la meta M' incluye a todos los ítems sobre los que actualmente no se satisface su submeta más todos aquellos que hay que visitar para que la submeta impuesta sobre algún concepto se cumpla. Observe cómo queda la última fila de la tabla 2 donde se describe la heurística h1.

$s_i = (K(i_1), K(i_2), ..., K(i_n))$. M' = $\{(i_k, et_{SEM}^{k}), ..., (c_s, et_{SEM}^{s}), ...\}$

Paso 1. Inicializar la ruta. r1 = {}.

Paso 2. Calcular la secuencia de visitas para llegar a la meta. r1:$s_i \rightarrow s_m$

Para cada submeta $(i_j, et_{SEM}^{j}) \in$ M' *hacer*

  Si $K(i_j)$ < valor-numérico(et_{SEM}^{j}) *entonces* r1 = r1 + {visit(i_j)}

Para cada submeta $(c_j, et_{SEM}^{j}) \in$ M' *hacer*

  Calcular $v(c_j)$ para el estado s_i y la submeta (c_j, et_{SEM}^{j}) (algoritmo 6)

  Para cada $i_k \in v(c_j)$ *hacer*

    Si visit(i_k) $\notin$ r1 *entonces* r1 = r1 + {visit(i_k)}

Paso 3. Obtener el costo de la ruta. $h1(s_i) = g(r1)$

En la heurística **h2** se obtiene, a partir de la anterior ruta, el mínimo conjunto de reglas de actualización que es necesario ejecutar para actualizar todos los ítems incluidos en ésta. La ruta estimada por h2 tendrá un número de visitas igual o menor al obtenido en h1, ya que aprovecha cuando en el cuerpo de una regla se actualizan varios ítems de golpe.

Concretamente, la ruta estimada en h2 está compuesta por los ítems cabeza de las reglas de actualización necesarias para actualizar todos los ítems que aún no satisfacen su submeta y/o que se requieren para satisfacer la submeta de un concepto. Observe cómo queda, después de las modificaciones oportunas, la última fila de la tabla 3 donde se describe la heurística h2.

$s_i = (K(i_1), K(i_2), ..., K(i_n))$. M' = $\{(i_k, et_{SEM}^{k}), ..., (c_s, et_{SEM}^{s}), ...\}$

Paso 1. Inicializar la ruta. r2 = {}.

Paso 2. Obtener un conjunto, $\mathbf{cj_m}$, con todos los ítems que no satisfacen su submeta en s_i o que son necesarios para que un concepto satisfaga su submeta.

  $cj_m = \varnothing$.

Para cada submeta $(i_j, et_{SEM}^j) \in M'$ *hacer*

si $K(i_j) <$ valor-numérico(et_{SEM}^j) *entonces* $cj_m = cj_m \cup \{i_j\}$

Para cada submeta $(c_j, et_{SEM}^j) \in M'$ *hacer*

Calcular $v(c_j)$ para el estado s_i y la submeta (c_j, et_{SEM}^j)

Para cada $i_k \in v(c_j)$ *hacer*

Si $i_k \notin cj_m$ *entonces* $cj_m = cj_m \cup \{i_k\}$

Paso 3. Obtener la ruta que partiendo desde s_i llega hasta s_m. r2: $s_i \rightarrow s_m$

Mientras $cj_m \neq \varnothing$ *hacer*

Buscar en el conjunto de reglas de actualización, la regla $Ru(i_u)$ que actualiza en su cuerpo el mayor número de ítems contenidos en cj_m.

$r2 = r2 + \{visit(i_u)\}$

Para cada ítem $i_k \in cj_m$, *si* Actualización$(i_k) \in Ru(i_u)$ *entonces* $cj_m = cj_m - i_k$

Paso 4. Devolver el costo de la ruta encontrada. $h2(s_i) = g(r2)$.

Cuando la heurística h2 se usa en la búsqueda A*, para que ésta sea admisible, se calcula el costo hasta la meta multiplicando el número de visitas en la ruta estimada, r2, por el costo más bajo de un ítem en la estructura de navegación actual (ecuación 15). Puesto que el número de visitas en la ruta estimada nunca excede al número de visitas en la ruta real se tiene que h2 es admisible.

Para que h2 siga siendo admisible tras las modificaciones realizadas, se debe cumplir que el número de visitas estimadas para satisfacer la meta de un concepto, $v(c_i)$, nunca exceda al número de visitas que realmente se necesitan. De acuerdo a esto, es evidente que **h2 sigue siendo admisible**, porque precisamente el conjunto $v(c_i)$ ha sido calculado de forma que contiene el menor número de ítems posibles. Es cierto, que puede existir otros conjuntos, $v'(c_i)$, con el mismo número de ítems que $v(c_i)$, pero nunca con menos, ya que en cada paso se elige el ítem que proporciona mayor ganancia de conocimiento sobre el concepto.

En el ejemplo anterior, existe un conjunto v'(Hatha-Yoga) = {I5, I6} que tiene dos ítems como el conjunto mínimo v(Hatha-Yoga) = {I5, I7}, y que satisface también la submeta impuesta sobre *Hatha-Yoga*, aunque obtiene para éste un conocimiento ligeramente menor.

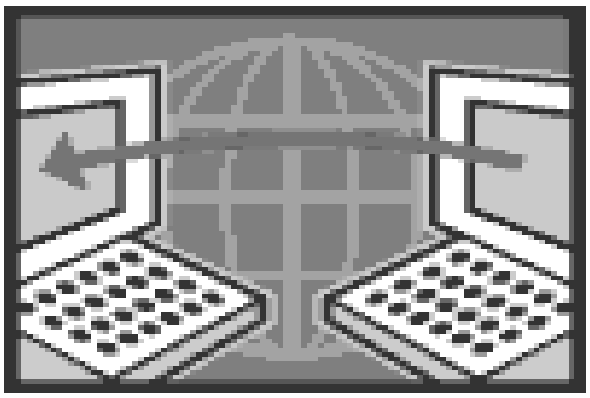
$s_k' = (0,3,2,4,\mathbf{4},\mathbf{4},\mathbf{0})$

$K(\text{Hatha-Yoga}) = 0{,}4 \times \mathbf{4} + 0{,}25 \times \mathbf{4} + 0{,}2 \times \mathbf{0} + 0{,}15 \times 1 = 2{,}75$ ("high") $\geq$ "high"

Aunque los otros conjuntos, $v'(c_i)$, pudieran presentar ítems con menor costo que los incluidos en $v(c_i)$, no repercute en la admisibilidad de h2, ya que en la búsqueda A* sólo afecta la longitud de la ruta estimada, pues para todos los ítems incluidos en ésta se asume el menor costo posible.

CAPÍTULO 21

Retroalimentación Adaptativa



Resumen

En este capítulo se describe y especifica una técnica denominada *Retroalimentación Adaptativa*, que estudia el comportamiento de los usuarios que navegan en modo libre (tradicional o por conceptos), y a partir de éste obtiene información sobre el uso de las estructuras de navegación, sus características positivas y negativas, y el grado en que éstas se superponen con las estrategias de navegación seguidas mayoritariamente por los usuarios.

Tabla de contenidos

Retroalimentación Adaptativa

1. INTRODUCCIÓN

La construcción de la estructura conceptual de memorización, EC_M, es un proceso de diseño evolutivo, durante el cual, el autor elimina, añade o modifica los elementos actualmente incluidos en ésta, para conseguir la representación que desea del dominio de conocimiento del sistema hipermedia.

Este desarrollo evolutivo es impulsado y dirigido por el propio autor de acuerdo a su criterio personal, y a menudo, es motivado por una o varias de las siguientes causas:

a) cambios en el dominio conceptual,

b) actualizaciones/ampliaciones en el dominio de información, o

c) transformaciones en el contexto del sistema.

En cualquier caso, el proceso de refinamiento incremental que da lugar a la EC_M final, es realizado por el autor haciendo uso de la capacidad evolutiva del modelo. Concretamente, éste es resuelto a través del conjunto de acciones evolutivas definidas en el Sistema de Memorización del modelo SEM-HP [García, 01c].

De forma adicional, el Sistema de Aprendizaje aporta la posibilidad de semi-automatizar este proceso de refinamiento en función de la navegación que realizan los usuarios del sistema. Con esto se consigue enriquecer el proceso de desarrollo evolutivo de la EC_M y de las presentaciones, EC_P^k, creadas a partir de ésta. De este modo, en la evolución de las estructuras, *se complementan las decisiones de diseño del autor con las modificaciones derivadas de la utilización del sistema* por parte de los usuarios.

Este **proceso de evolución** es **semiautomático**, ya que, aunque es impulsado por los usuarios y sugerido por el propio sistema, debe ser aceptado y dirigido por el autor, que tiene la última palabra. Es decir, es éste quien decide si se realizan o no sobre las estructuras pertinentes aquellos cambios identificados como necesarios durante su uso.

Puesto que la navegación que hace un usuario en el sistema está condicionada por el modo de navegación que realiza (capítulo 15), no todos los usuarios son susceptibles de análisis. Concretamente, sólo debe ser examinada la navegación que realizan los "usuarios libres", es decir aquellos que visitan la información del sistema hipermedia, sin necesidad de satisfacer ninguna restricción previamente impuesta por el autor.

De esta forma, el análisis aísla perfectamente las decisiones de navegación de los usuarios de los caminos de navegación trazados por el autor. Dentro de estos llamados "usuarios libres" se incluyen tanto los que visitan ítems siguiendo el esquema tradicional de navegación como los que navegan, también sin restricciones de acceso, a través de resúmenes de conceptos.

2. BENEFICIOS DE UN MECANISMO DE RETROALIMENTACIÓN

En resumen, cuando un usuario visita información utilizando un modo de navegación libre, ya sea tradicional (capítulo 15) o por conceptos (capítulo 16), el sistema de aprendizaje estudia y analiza su comportamiento. Para después combinarlo con la conducta de otros usuarios de su mismo tipo y *averiguar así las estrategias de navegación que se utilizan mayoritariamente*.

Finalmente, basándose en los patrones de navegación de los usuarios libres, el sistema es capaz de sugerir al autor una serie de modificaciones sobre el principal modelo de representación del sistema hipermedia, esto es, la estructura conceptual de memorización y sobre las presentaciones obtenidas a partir de ésta. Siguiendo estas sugerencias, el autor puede adaptar los modelos de representación del sistema hipermedia a los modelos mentales que de ellos tienen los usuarios del mismo.

Consideramos este mecanismo de evolución semiautomático un **proceso de retroalimentación**, donde las estructuras del sistema hipermedia se redefinen en función de la navegación que sobre ellas realiza un grupo significativo de usuarios (figura 1). Además, se trata de un **proceso adaptativo**, ya que permite aproximar los modelos de representación del sistema a la concepción que de éstos tienen sus usuarios.

> **Def 21.1 [Retroalimentación adaptativa]** Nombramos retroalimentación adaptativa a la técnica empleada por el Sistema de Aprendizaje para: 1) analizar e integrar el comportamiento navegacional de los usuarios que recorren el sistema hipermedia en modo libre, 2) cotejar la estructuras trazadas por éstos con las estructuras definidas previamente por el autor y 3) sugerir las modificaciones necesarias para hacer converger ambas estructuras.

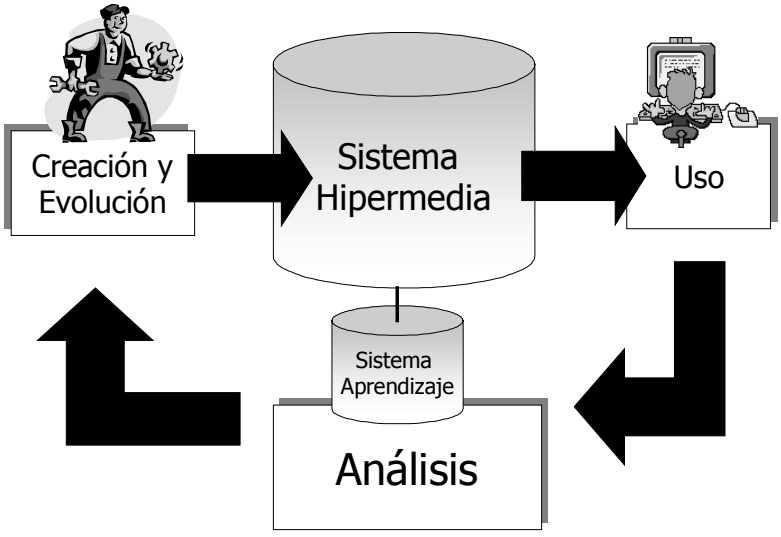


Figura 1: Retroalimentación adaptativa

La aplicación de esta técnica tiene tres beneficios, claros e inmediatos, sobre el uso del sistema hipermedia:

- En cierta medida, el mecanismo de retroalimentación implica un proceso de **adaptación al grupo de usuarios**, ya que hace que las estructuras de navegación

disponibles en el sistema converjan hacia la representación mental que el grupo de usuarios tiene sobre su contenido.

- Obviamente, un usuario se encuentra más cómodo usando un sistema hipermedia cuya estructura contempla el dominio conceptual y de información de manera afín a su concepción mental sobre éstos. Esto conlleva una reducción general de los problemas de navegación, particularmente de las situaciones de desorientación y pérdida.

- Puesto que las estructuras redefinidas a partir del proceso de retroalimentación son proporcionadas también a los nuevos usuarios, el conocimiento del grupo de usuarios sirve para aconsejar a futuros usuarios. De modo que los usuarios novatos se benefician del proceso de adaptación al grupo, aprendiendo de las estrategias de navegación de los más veteranos [Bollen, 98] [Bollen, 00].

Antes de entrar de lleno en la descripción formal del mecanismo de retroalimentación adaptativo, donde además se detallan las diferentes utilidades del mismo, conviene dar una visión más concreta del tipo de modificaciones que esta técnica es capaz de sugerir.

Entre éstas podemos destacar la capacidad de descubrir asociaciones conceptuales aceptadas por la mayoría de los usuarios, que sin embargo, no han sido incluidas por el autor en la estructura conceptual de memorización, o identificar conceptos, ítems y relaciones conceptuales definidas por el autor, que raramente son consideradas por los usuarios del sistema.

Supongamos que la mayoría de los usuarios que recorren la estructura de navegación que se muestra en la figura 2.1, siempre visitan el ítem *I1* o *I2* después de visitar el ítem *I4*. Esto quiere decir que implícita o explícitamente aceptan una asociación conceptual entre los conceptos *Ha* y *Loto*, cuya inserción es sugerida al autor.

La estructura de navegación que incorpora esta recomendación se muestra en la figura 2.2, donde se resalta con mayor grosor la relación conceptual finalmente incorporada. Obviamente etiquetar semánticamente dicha relación es tarea del autor.

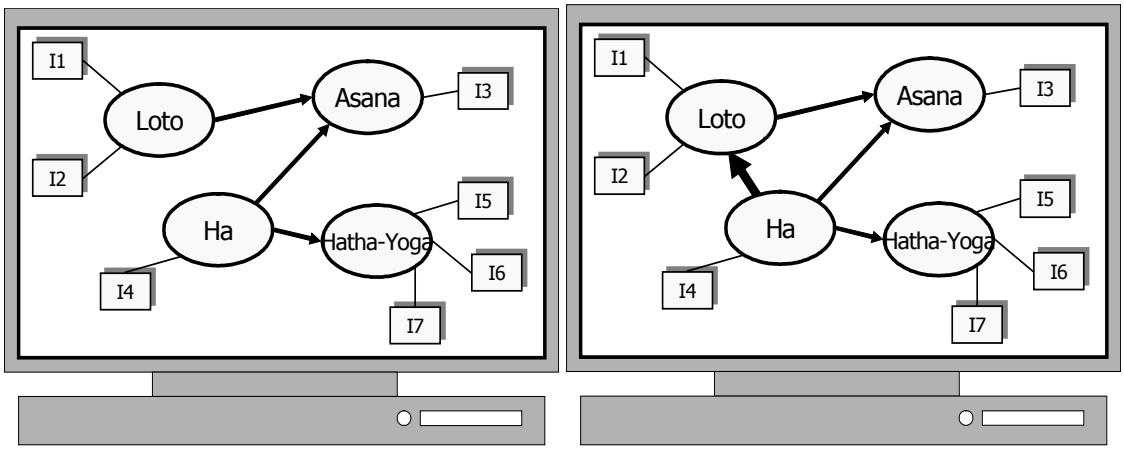


Figura 2.1: Ejemplo de adaptación por retroalimentación**: Figura 2.2**

3. RETROALIMENTACIÓN EN LA NAVEGACIÓN TRADICIONAL

Tal y como establece Johan Bollen [Bollen, 00], entendemos que si los modelos de los usuarios y de los diseñadores se superponen de forma pobre, esto es, tienen una visión distinta de la realidad modelada, la interacción del usuario con el sistema va a ser ineficiente.

Por lo tanto, es conveniente proporcionar un mecanismo de realimentación que permita a los usuarios establecer, de alguna forma, las relaciones que conciben entre los conceptos. Por supuesto, no obligamos al usuario a declarar explícitamente las relaciones, que a su juicio, existen entre los conceptos, sino que aprovechamos las relaciones implícitas que se desprenden de su proceso de navegación.

La estructura de navegación proporcionada durante la navegación tradicional coincide con la red semántica usada para representar una determinada estructura conceptual de presentación, EC_P^k. El objetivo principal de la retroalimentación adaptativa es obtener las asociaciones que los usuarios admiten entre los conceptos mostrados en dicha presentación.

Para ello, si la mayoría de los usuarios de una presentación que se encuentran visitando algún ítem asociado funcionalmente al concepto c_i justo después acceden a un ítem ligado a c_j, el sistema interpreta que consciente o inconscientemente la mayoría de ellos acepta una relación semántica (r_c) entre ambos conceptos.

> **Def 21.2 [Matriz de transiciones]** Denominamos matriz de transiciones (MT) a la estructura de datos utilizada para identificar las asociaciones conceptuales que el grupo de usuarios tiene en mente durante su navegación por una red semántica.

Concretamente, la matriz de transiciones asociada a una presentación EC_P^k es notada $\mathbf{MT_p^k}$ y almacena para cada pareja de conceptos, c_i, c_j, incluidos en su dominio conceptual, $c_i \in C(EC_P^k)$ y $c_j \in C(EC_P^k)$, información sobre la existencia de una relación r_c: $c_i \rightarrow c_j$ en el esquema mental de quienes la recorren[1].

El sistema construye una matriz MT_p^k para cada presentación definida a partir de la EC_M. Pero, además, combinando estas matrices parciales, el sistema obtiene una matriz de transiciones correspondiente a la EC_M completa. Esta matriz se denomina $\mathbf{MT_m}$ y almacena información a cerca de la existencia o inexistencia, para los usuarios, de una relación semántica entre cada dos conceptos de la EC_M.

Ambas matrices son posteriormente examinadas con objeto de identificar las diferencias más significativas entre las estructuras definidas por el autor y la concepción que de éstas tienen sus usuarios.

[1] Usamos la notación r_c: $c_i \rightarrow c_j$ en lugar de $<c_i, r_c, c_j>$ para que quede más claro el sentido de la relación.

3.1 Matriz de transiciones de una presentación (MT_p^k)

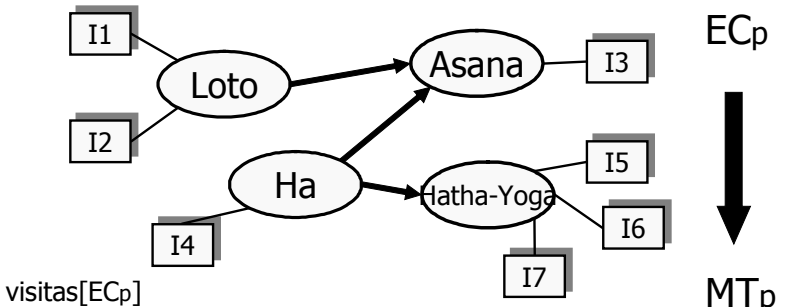
Def 21.3 [MT_p^k] La matriz de transiciones asociada a una presentación tiene una fila y una columna por cada concepto incluido en ésta. Así, siendo $C(EC_P^k)=\{c_1, c_2, ..., c_n\}$ el conjunto de conceptos mostrados en la presentación EC_P^k, la matriz MT_p^k asociada es una matriz cuadrada de orden *n*, que contiene por tanto $n \times n$ elementos.

La celda $MT_p^k[c_i, c_j]$ de la matriz MT_p^k representa el número de veces que un usuario de la presentación EC_p^k ha seguido una relación conceptual entre ambos conceptos, concretamente desde el concepto origen c_i hasta el concepto destino c_j (r_c: $c_i \rightarrow c_j$).

Si existen varias relaciones conceptuales entre dos conceptos c_i y c_j, es indiferente la relación que se sigue para que se incremente el valor de la celda $MT_p^k[c_i, c_j]$. Cabe resaltar que no sólo se tienen en cuenta las relaciones conceptuales físicas, es decir, aquellas que aparecen en la presentación, sino también las relaciones conceptuales que sin existir materialmente son seguidas por el usuario al moverse de un concepto a otro durante su navegación.

Inicialmente, toda matriz de transiciones es una matriz nula o matriz cero, ya que todas sus celdas se inicializan con el valor 0. Después, cada vez que un usuario visita un ítem en una presentación EC_p^k se incrementa en 1 el valor de una determinada celda de su matriz de transiciones, MT_p^k. Concretamente, el valor de la celda $MT_p^k[c_i, c_j]$ se incrementa ($MT_p^k[c_i, c_j] = MT_p^k[c_i, c_j] +1$) cuando el usuario visita un ítem asociado al concepto c_j, y el último ítem que visitó estaba ligado al concepto c_i.

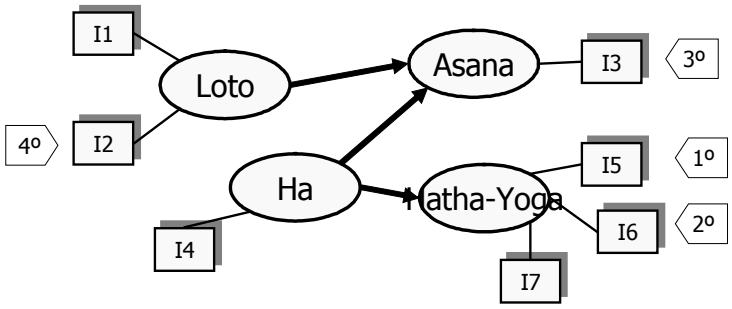
La figura 3 muestra la matriz de transiciones para la presentación del ejemplo en su estado inicial (figura 3a) y después de que el lector haya visitado cuatro ítems en el siguiente orden: primero I5, después I6, I3 en tercer lugar y finalmente I2 (figura 3b).



0	Loto	Ha	Asana	Hatha
Loto	0	0	0	0
Ha	0	0	0	0
Asana	0	0	0	0
Hatha	0	0	0	0

$MT_p[Asana, Ha]$

Figura 3a: MT_p inicial



4	Loto	Ha	Asana	Hatha
Loto	0	0	0	0
Ha	0	0	0	0
Asana	1	0	0	0
Hatha	0	0	1	2

MT_p'

Figura 3b: MT_p después de cuatro visitas

Observe como la matriz inicialmente a cero modifica el valor de tres de sus celdas tras la navegación del usuario, lo cual significa que durante esa navegación el usuario ha seguido tres relaciones conceptuales: r1: Hatha → Hatha, r2: Hatha → Asana y r3: Asana → Loto.

Obviamente, la representación de las matrices de transición en la figura 3 no es formal. Sin embargo, usaremos durante este capítulo dicha representación por ser más ilustrativa (ver matrices equivalentes en la ecuación 1).

$$MT_p = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow MT_p' = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix} \qquad \textbf{(1)}$$

3.2 Información extraída de MT_p^k

El valor situado en cada celda de una matriz de transiciones MT_p refleja el seguimiento, en esa presentación, de una determinada relación conceptual. De estas relaciones sólo nos interesan las que impliquen dos conceptos diferentes, ya que la de un concepto consigo mismo estará motivada en la mayoría de los casos, no por una relación efectiva, sino por el hecho de que un usuario accede sucesivamente a varios ítems del concepto con la intención de aprender más sobre éste. Por este motivo, los elementos de la diagonal principal son ignorados en el procedimiento de análisis que se describe a continuación.

3.2.1 Contador de visitas de una presentación

El número total de visitas realizadas a ítems de una presentación puede calcularse sumando el contenido de todas las celdas de su matriz de transiciones, tal y como se especifica en la ecuación 2. Obviamente sólo estamos contabilizando las visitas realizadas durante la navegación tradicional.

$$visitas\left(EC_p^k\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} MT_p^k[c_i, c_j] \qquad (2)$$

Comparando este **contador de visitas global** en las distintas presentaciones definidas a partir de una misma estructura conceptual de memorización, el sistema puede inferir qué presentaciones son más visitadas y reflejar el tipo de usuarios que usan el sistema. Asimismo puede identificar presentaciones posiblemente prescindibles, es decir, aquellas con un número extremadamente bajo de visitas.

Esta información se proporciona al autor a través de una lista de presentaciones ordenada de mayor a menor número de visitas. De modo que la posición en la clasificación de una presentación EC_p^k es posterior a la posición de otra presentación EC_p^r si se tiene que: visitas(EC_p^k) < visitas(EC_p^r).

A modo informativo, junto a cada presentación se muestra el número total de visitas realizadas sobre sus ítems y el subdominio o subdominios de conocimiento capturados. De esta forma, el autor puede hacerse una idea bastante acertada de las parcelas de conocimiento más navegadas por los usuarios, advirtiendo así sus intereses al respecto.

De modo análogo, observando el etiquetado de las últimas presentaciones de la lista, el autor puede determinar qué presentaciones poseen unas características adecuadas para sólo unos pocos usuarios y plantearse la posibilidad de modificarlas o incluso eliminarlas.

3.2.2 Índice de visitas de una presentación

Para poder sacar conclusiones de una forma más fácil y cómoda, el sistema proporciona al autor una serie de información de análisis inferida a partir del conjunto de presentaciones existentes y el contador de visitas de cada una de éstas.

Para cada presentación EC_P^k se calcula un índice, denominado **Ivisitas(EC_P^k)**, que refleja si el número de visitas realizadas sobre dicha presentación está por encima o por debajo de la media teórica y en qué grado.

La *media teórica* representa el caso en que las visitas se reparten equitativamente entre las presentaciones existentes, y por lo tanto, se calcula dividiendo el número total de visitas entre el número de presentaciones. El número total de visitas es la suma de las visitas totales realizadas en cada presentación. La ecuación 3 muestra cómo se obtiene dicho índice.

$$Ivisitas\left(EC_p^k\right) = \frac{visitas\left(EC_p^k\right)}{\left[\frac{\sum_{j=1}^{r} visitas\left(EC_p^j\right)}{r}\right]} \quad \text{siendo } r \text{ el número de presentaciones} \qquad (3)$$

A partir de la fórmula anterior se deduce que:

a) el indicador Ivisitas(EC_p^k) vale 1 si las visitas realizadas sobre la presentación EC_p^k coincide con la media teórica,

b) es menor que 1 si está por debajo de la media, siendo 0 en el peor caso, esto es cuando no se ha visitado la presentación (visitas(EC_p^k) = 0),

c) es mayor que 1 si supera la media teórica, siendo *r* en el mejor caso, esto es cuando todas las visitas se han realizado sobre esa presentación: $\text{visitas}\left(EC_{p}^{k}\right) = \sum_{j=1}^{r} visitas\left(EC_{p}^{j}\right)$.

Para determinar el conjunto de presentaciones más utilizadas, al que notaremos $\mathbf{P^+}$, se toman aquellas presentaciones cuyo indicador de visitas es superior a un determinado umbral v_{sup}. Por el contrario, el conjunto de presentaciones menos utilizadas, notado $\mathbf{P^-}$, incluirá aquellas presentaciones con un indicador de visitas menor que un umbral inferior $v_{inf.}$ (véase la ecuación 4).

$$EC_p^k \in P^+ \text{ si } Ivisitas(EC_p^k) > v_{sup} \qquad (4)$$
$$EC_p^k \in P^- \text{ si } Ivisitas(EC_p^k) < v_{inf}$$

Por defecto, el valor de v_{sup} y v_{inf} es 1, lo cual indica que se incluyen en P^+ las presentaciones cuyo número de visitas supera la media teórica y en P^- las que están por debajo de ésta. Sin embargo, el autor puede reestablecer ambos valores, siempre que cumpla las siguientes restricciones: El umbral superior, v_{sup}, debe ser un número real mayor o igual que 1 y menor que el número de presentaciones (*r*). Y el umbral inferior, v_{inf}, debe ser un número real mayor que 0 y menor o igual que 1. Esto es, $v_{sup} \in [1,r)$ y $v_{inf} \in (0,1]$.

3.2.3 Análisis de las presentaciones más y menos visitadas

Sobre el conjunto $\mathbf{P^+}$ el sistema obtiene el perfil, en cuanto al subdominio de conocimiento se refiere, de las presentaciones más visitadas. Esta información puede ayudar al autor a conocer las características de las presentaciones con mayor número de visitas. Aprendiendo así un poco más acerca de los usuarios de su sistema, con el fin de refinar esas presentaciones, añadir presentaciones parecidas o definir nuevas estructuras de aprendizaje a partir de ellas.

Concretamente la información proporcionada, cuya obtención se muestra en la tabla 1, es la siguiente:

a) El conjunto de **subdominios de conocimiento** capturados *en todas las presentaciones incluidas en* P^+, y

b) El conjunto de subdominios de conocimiento capturados *en alguna presentación de* P^+.

Tabla 1: Subdominio de conocimiento de las presentaciones más visitadas

Perfil de P^+ para el subdominio de conocimiento[2]	
$\forall_{k:1..m}$ tal que $EC_P^k \in P^+$ entonces listaSubC(EC_P^k) = subdominio-de-conocimiento(EC_P^k)	
Intersección **(a)**	listaSubC(EC_P^1) ∩ listaSubC(EC_P^2) ∩ ... ∩ listaSubC(EC_P^m)
Unión **(b)**	listaSubC(EC_P^1) ∪ listaSubC(EC_P^2) ∪ ... ∪ listaSubC(EC_P^m)

[2] listaSubC(EC_P^k) y subdominio-de-conocimiento(EC_P^k) son equivalentes, se requiere y utiliza la primera notación simplemente por cuestión de espacio.

En este caso la intersección (a) es bastante significativa, ya que permite al autor detectar el subdominio o subdominios de conocimiento que interesan colectivamente a los usuarios del sistema.

Supongamos el conjunto P^+ formado por tres presentaciones EC_p^1, EC_p^2 y EC_p^3, cuyo valor en la etiqueta subdominio-de-conocimiento se muestra en la tabla 2.

Tabla 2: Subdominio de conocimiento de tres presentaciones en P^+

	subdominio-de-conocimiento(EC_P^k) ≡ listaSubC(EC_P^k)
EC_P^1	{("adaptación al usuario", 75%), ("modelo SEM-HP", 40%)}
EC_P^2	{("adaptación al usuario", 95%)}
EC_P^3	{("adaptación al usuario", 80%), ("evolución", 30%)}

En este caso, la unión de los subdominios de conocimiento (b) es *{"adaptación al usuario", "modelo SEM-HP", "evolución"}*, lo cual indica que esos tres subdominios interesan a un grupo elevado de usuarios.

Por su parte, la intersección de subdominios (a) centra el interés del usuario en el subdominio *"adaptación al usuario"* ya que éste es el único que aparece en todas las presentaciones más visitadas.

Sobre el conjunto $\mathbf{P}^-$ se obtiene también la anterior información, de modo que el autor puede observar el conjunto unión e intersección de los subdominios incluidos en las presentaciones menos visitadas. Esto le permite identificar los **subdominios de conocimiento** que *no interesan*, al menos mayoritariamente, a los usuarios de su sistema.

A partir de dicha información, el autor puede plantearse eliminar o modificar determinadas presentaciones. Sin embargo, puede que el motivo por el que una presentación EC_P^k apenas se utiliza, no se encuentre en la parcela de conocimiento que representa, sino en que no existen estructuras de aprendizaje EC_A^j definidas a partir de ella con un rango de experiencia adecuado.

Es decir, a la hora de elegir la estructura de aprendizaje que se proporciona a un usuario, interviene el subdominio de interés de éste, pero también el grado de experiencia que posee en la navegación hipermedia y en la materia del sistema actual. Con lo cual, aunque una presentación EC_P^k incluya el subdominio deseado, puede no ser ofrecida al usuario si ninguna de las EC_A^j definidas sobre ésta se ajustan a su experiencia.

Así, para cada una de las presentaciones incluidas en P^- se hace un análisis que advierte sobre la **experiencia** a la que están orientadas las estructuras de aprendizaje que actualmente existen sobre ella. Concretamente, para cada $EC_P^k \in P^-$ se indica:

a) El intervalo de experiencia común a todas las estructuras de aprendizaje definidas sobre EC_P^k, esto es su *intersección*.

b) El intervalo de experiencia que engloba a cada uno de los intervalos que etiquetan una estructura de aprendizaje definida sobre la presentación EC_P^k, es decir la *unión*.

c) El intervalo de experiencia *medio* en las estructuras de aprendizaje definidas sobre EC_P^k.

Por supuesto, este análisis se realiza de forma separada para la experiencia de navegación (tabla 3) y la experiencia en la materia (tabla 4).

Tabla 3: Experiencia de navegación de las presentaciones menos visitadas

Perfil de una $EC_P^k \in P^-$ para la experiencia de navegación	
$\forall_{j:1..m}$ tal que $EC_A^j = (EC_M, Rw, MU, \mathbf{EC_P^k}, RTnb, Ro, Ru, Rk)$ entonces $Intervalo_{ExpN}(EC_A^j) = experiencia\text{-}navegación(EC_A^j) = [et1_{SEM}^j, et2_{SEM}^j]$	
Intersección **(a)**	$Intervalo_{ExpN}(EC_A^1) \cap Intervalo_{ExpN}(EC_A^2) \cap ... \cap Intervalo_{ExpN}(EC_A^m)$
Unión **(b)**	$Intervalo_{ExpN}(EC_A^1) \cup Intervalo_{ExpN}(EC_A^2) \cup ... \cup Intervalo_{ExpN}(EC_A^m)$
Media **(c)**	$\left[\frac{\sum_{j=1}^{m} valor\ -numérico\ \left(et1_{SEM}{}^{j}\right)}{m}, \frac{\sum_{j=1}^{m} valor\ -numérico\ \left(et2_{SEM}{}^{j}\right)}{m}\right]$

Tabla 4: Experiencia en la materia de las presentaciones menos visitadas

Perfil de una $EC_P^k \in P^-$ para la experiencia en la materia	
$\forall_{j:1..m}$ tal que $EC_A^j = (EC_M, Rw, MU, \mathbf{EC_P^k}, RTnb, Ro, Ru, Rk)$ entonces $Intervalo_{ExpM}(EC_A^j) = experiencia\text{-}materia(EC_A^j) = [et1_{SEM}^j, et2_{SEM}^j]$	
Intersección **(a)**	$Intervalo_{ExpM}(EC_A^1) \cap Intervalo_{ExpM}(EC_A^2) \cap ... \cap Intervalo_{ExpM}(EC_A^m)$
Unión **(b)**	$Intervalo_{ExpM}(EC_A^1) \cup Intervalo_{ExpM}(EC_A^2) \cup ... \cup Intervalo_{ExpM}(EC_A^m)$
Media **(c)**	$\left[\frac{\sum_{j=1}^{m} valor\ -numérico\ \left(et1_{SEM}{}^{j}\right)}{m}, \frac{\sum_{j=1}^{m} valor\ -numérico\ \left(et2_{SEM}{}^{j}\right)}{m}\right]$

En este caso, la unión (b) es especialmente interesante. Si para una presentación EC_P^k se obtiene un intervalo de experiencia unión que no abarca todos los grados posibles, esto es, difiere del intervalo ["nulo", "total"], significa que no existen estructuras de aprendizaje definidas sobre esa presentación que sean adecuadas para un usuario cuyo grado de experiencia no se encuentra en la unión.

La situación expuesta justifica el número bajo de visitas a la presentación, más cuanto mayor es el número de grados excluidos, por lo que la solución del autor seguramente no será eliminar la presentación sino definir para ésta nuevas estructuras de aprendizaje que abarquen los grados de experiencia marginados.

Supongamos ahora una presentación EC_p^4 incluida en P^-. A partir de ésta se definen tres estructuras de aprendizaje EC_A^1, EC_A^2 y EC_A^3, cuyo etiquetado respecto a la experiencia se contempla en la tabla 5.

En la tabla 6 se muestra la unión, la intersección y la media de los intervalos de experiencia que etiquetan las tres estructuras de aprendizaje definidas para la presentación EC_p^4.

Tabla 5: Etiquetado de las estructuras de aprendizaje definidas sobre EC_P^4

	experiencia-navegación(EC_A^j) ≡ $Intervalo_{ExpN}(EC_A^j)$	experiencia-materia(EC_A^j) ≡ $Intervalo_{ExpM}(EC_A^j)$
EC_A^1	[“nulo”, “medio”]	[“medio”,”alto”]
EC_A^2	[“nulo”, “total”]	[“bajo”, “alto”]
EC_A^3	[“alto”,”total”]	[“medio”, “total”]

Tabla 6: Perfil de experiencia para $EC_P^4 \in P^-$

	experiencia-navegación	experiencia-materia
Unión	[“nulo”, “total”]	[“bajo”,”total”]
Intersección	[∅]	[“medio”, “alto”]
Media	[3/3, 10/3] ≈ [“bajo”, “alto”]	[5/3, 10/3] ≈ [“medio”, “alto”]

Como puede observar, el intervalo unión para el atributo experiencia-materia es [“bajo”, “total”] lo que indica que no existen estructuras de aprendizaje asociadas a EC_P^4 que estén orientadas a usuarios con experiencia “nula” en la materia. Ante este dato, el autor puede definir sobre la presentación EC_P^4 nuevas estructuras de aprendizaje, una o varias, destinadas a los usuarios que poseen precisamente ese grado de experiencia en la materia.

Por supuesto, estas técnicas de análisis, como todas las que se detallan en este capítulo, únicamente son relevantes cuando el uso del sistema proporciona una muestra considerable. Además, la utilidad de la información proporcionada queda a expensas de la interpretación del autor y las modificaciones que en consecuencia éste lleve a cabo.

3.3 Matriz de transiciones de memorización (MT_m)

La matriz de transiciones de una presentación refleja las estrategias de navegación de sus usuarios a través de las relaciones conceptuales seguidas por éstos durante el proceso de navegación tradicional que realizan. Por lo tanto, podemos decir que la matriz de transiciones de una presentación, MT_P^k, captura el modelo mental que de dicha presentación, EC_P^k, tienen los usuarios que seleccionan libremente los ítems de su dominio de información.

Sin embargo, el modelo mental capturado en esta matriz de transiciones, MT_P^k, es un modelo parcial, en el sentido de que se limita a una parcela concreta del dominio de conocimiento del sistema hipermedia. Para obtener el modelo mental que estos usuarios poseen sobre la estructura conceptual de memorización es necesario superponer los modelos parciales obtenidos para cada subdominio.

Def 21.4 [MT_m] La matriz de transiciones de una estructura conceptual de memorización EC_M se nota como MT_m y contiene las relaciones definidas entre cada dos conceptos por el conjunto de usuarios que recorren cualquiera de sus

presentaciones en modo tradicional. Concretamente, para calcular la matriz de transiciones MT_m, el sistema combina las matrices de transiciones, MT_p^1, MT_p^2,..., MT_p^r, de las *r* presentaciones EC_P^1, EC_P^2,..., EC_P^r definidas a partir de EC_M.

La matriz de transiciones MT_m tiene una fila y una columna por cada concepto incluido en la EC_M. Sea $C(EC_M) = \{c_1, c_2, ..., c_s\}$ el conjunto de los *s* conceptos existentes en la estructura conceptual de memorización EC_M, la matriz asociada MT_m es una matriz cuadrada de orden *s*, es decir, una matriz con dimensión $s \times s$.

Concretamente, la matriz *MT_m se construye sumando las matrices de transiciones MT_p^k* de todas las presentaciones EC_P^k creadas a partir de la EC_M. Puesto que una presentación no tiene por qué mostrar todos los conceptos de la estructura conceptual de memorización (de hecho, lo normal es que no sea así), en el cálculo de la celda $MT_m[c_i,c_j]$ sólo intervienen las matrices de las presentaciones que incluyen ambos conceptos y para las que, por lo tanto, existe en su matriz de transiciones una celda $MT_p^k[c_i, c_j]$ asociada a dicha pareja de conceptos.

Aquellas celdas de la matriz $MT_m[c_i, c_j]$ para cuyo cálculo no existe ninguna celda $MT_p^k[c_i, c_j]$ se marcan con una *X*, indicando que los conceptos c_i y c_j no se han incluido juntos en ninguna presentación de la EC_M. Véase el cálculo de MT_m en la ecuación 5.

$$MT_m[c_i, c_j] = \sum_{k=1}^{r} MT_p^k\left[c_i, c_j\right] \text{ tal que } c_i \in C(EC_P^k) \wedge c_j \in C(EC_P^k) \qquad (5)$$

$$MT_m[c_i, c_j] = X \text{ si } \neg\exists EC_P^k \text{ tal que } c_i \in C(EC_P^k) \wedge c_j \in C(EC_P^k)$$

3.3.1 Contador de visitas

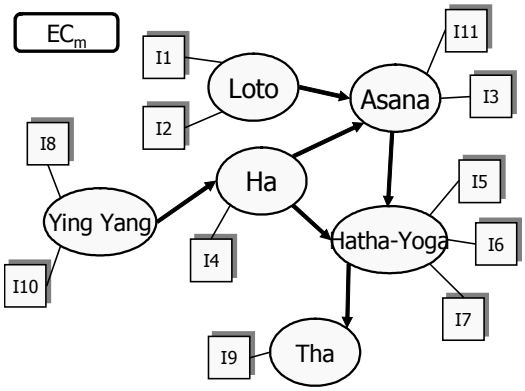
El número total de visitas a ítems de la EC_M es la suma de las visitas totales realizadas en modo tradicional sobre cada una de las presentaciones EC_P^1, EC_P^2,..., EC_P^r creadas a partir de ella. Obviamente, este mismo valor se obtiene si sumamos todas las celdas distintas de *X* en su matriz de transiciones MT_m (véase ecuación 6).

$$\textbf{visitas}(\mathbf{EC_M}) = \sum_{k=1}^{r} \text{visitas}(EC_P^k) = \sum_{i=1}^{s}\sum_{j=1}^{s} MT_m\left[c_i, c_j\right] \; tal \; que : MT_m\left[c_i, c_j\right] \neq X \qquad (6)$$

La utilidad principal de este dato, visitas(EC_M), es ayudar al autor a hacerse una idea aproximada de la utilización, no de una presentación particular, sino de todo el sistema hipermedia.

La figura 4 muestra la matriz de transiciones $MT_m[6\times6]$ creada para la estructura conceptual de memorización que se toma como ejemplo. Observe que en la MT_m existe una fila y una columna etiquetada con cada uno de los seis conceptos incluidos en EC_M, esto es *Loto*, *Ha*, *Asana*, *Hatha* (*Hatha-Yoga*), *Y-Y*(*Ying-Yang*) y *Tha*.

El contenido de cada celda en MT_m es obtenido, de acuerdo a la ecuación 5, a partir de las matrices de transiciones $MT_p^1[4\times4]$ y $MT_p^2[3\times3]$ correspondientes a las dos únicas presentaciones definidas sobre la EC_M.



159	Loto	Ha	Asana	Hatha
Loto	4	5	10	10
Ha	15	12	16	5
Asana	5	6	8	20
Hatha	5	25	5	8

102	Ha	Hatha	Tha
Ha	2	15	10
Hatha	5	15	20
Tha	20	10	5

MT_p

MT_m

261	Loto	Ha	Asana	Hatha	Y-Y	Tha
Loto	4	5	10	10	X	X
Ha	15	14	16	20	X	10
Asana	5	6	8	20	X	X
Hatha	5	30	5	23	X	20
Y-Y	X	X	X	X	X	X
Tha	X	20	X	10	X	5

Figura 4: Matriz de transiciones de una EC_M

El contador de visitas en MT_m, nos indica que hasta el momento, diferentes usuarios han realizado un total de 261 visitas a ítems del sistema hipermedia siguiendo el modo de navegación tradicional. Observe que este contador coincide con la suma de todas las celdas en MT_m, pero además es la suma de los contadores de visitas en MT_p^1 y MT_p^2, 159 y 102 respectivamente.

3.3.2 Relaciones imposibles y conceptos olvidados

El uso de la marca *X* en la MT_m permite diferenciar claramente las **relaciones** conceptuales, $r_c{:}c_o \rightarrow c_d$, que no se siguen porque no cuadran en el esquema mental de los usuarios, de aquellas que no se siguen sencillamente porque son **imposibles**. Esto es, si el autor no ha incluido juntos en ninguna presentación los conceptos c_o y c_d, no hay lugar de que el usuario visite, sobre una misma estructura, primero un ítem asociado a c_o y luego uno asociado a c_d.

Del análisis de la matriz de transiciones MT_m de la figura 4, se deduce que la relación entre los conceptos *Asana* y *Loto* es poco aceptada por los usuarios estudiados, ya que el valor de la celda MT_m[Asana, Loto] tiene un valor relativamente pequeño (5 de 261 visitas). En caso de ser 0 indicaría que no es aceptada por ninguno de los usuarios analizados.

Sin embargo, la existencia de una *X* en la celda MT_m[Asana, Tha] plantea la imposibilidad de saber si esa relación es o no aceptada por los usuarios, ya que en ninguna de las dos presentaciones existentes se han incluido ambos conceptos.

De modo similar, la marca *X* nos permite detectar lo que llamamos **conceptos olvidados**, esto es, conceptos definidos en la estructura conceptual de memorización que sin embargo no se han incluido en ninguna presentación. La manera de identificar estos conceptos es inmediata, basta con que la fila y la columna asociada a un concepto en la matriz MT_m tenga todas sus celdas igual a *X*, para dictaminar que ese concepto no aparece en ninguna presentación EC_P^k de la EC_M (véase la ecuación 7).

$$\forall_{k:1..r}\, c_i \notin EC_P^k \;\text{ si }\; \forall_{j:1..s}\, MT_m[c_i, c_j] = X \;\wedge\; MT_m[c_j, c_i] = X \qquad (7)$$

Obviamente todas las relaciones que impliquen, como concepto origen o destino, a un concepto olvidado son relaciones imposibles.

En el ejemplo de la figura 4, aunque existen varias relaciones imposibles, el único concepto olvidado es *Ying-Yang*, ya que $\forall_{j:\,1..6}\ MT_m[\text{Ying-Yang}, c_j] = MT_m[c_j, \text{Ying-Yang}] = X$.

Cuando se notifica al autor la existencia de conceptos olvidados, este deberá reflexionar sobre si:

a) ese concepto es prescindible y tal caso borrarlo de la EC_M,

b) si olvidó incluirlo en alguna presentación existente o

c) si debe crear una nueva presentación que muestre ese concepto.

3.3.3 Conceptos visitados

Sobre los conceptos no olvidados, el autor puede estar interesado en conocer el número de visitas llevadas a cabo, quizá con el objetivo de identificar los conceptos más solicitados por los usuarios. O detectar, conceptos poco visitados, posiblemente por tener asociado un conjunto reducido o inapropiado de ítems.

Para conocer este dato, el número de visitas en torno a un concepto, el sistema no tiene más que sumar las celdas de su fila o su columna en la matriz de transiciones de la EC_M, tal y como se especifica en la ecuación 8.

$$\textbf{visitas}_i(c_i) \approx \sum_{j=1}^{s} MT_m[c_i, c_j] \approx \sum_{j=1}^{s} MT_m[c_j, c_i] \qquad (8)$$

De esta forma, las visitas totales realizadas sobre un concepto[3], $visitas_i(c_i)$, se obtienen sumando las visitas realizadas a cualquier ítem asociado al concepto en la EC_M, considerando para dicho cálculo el valor 0 cuando la celda está marcada con *X*.

Una vez obtenido el número de visitas a cada concepto, el sistema proporciona al autor una lista ordenada desde los conceptos más visitados hasta los conceptos olvidados. Junto a cada concepto se indica el número exacto de visitas realizadas a sus ítems.

Aplicando la ecuación 8 sobre la MT_m de la figura 4 se obtiene que las visitas realizadas sobre cada concepto son las siguientes:

	Hatha	**Ha**	**Asana**	**Tha**	**Loto**	**Y-Y**
$visitas_i(c_i)$	83	75	39	35	29	**0**

[3] Se ha utilizado la notación $visitas_i(c_i)$ en lugar de $visitas(c_i)$, porque este último es el nombre del atributo del modelo de usuario que registra el número de veces que un usuario concreto ha solicitado el resumen de c_i durante la navegación por conceptos.

En esta lista ordenada queda patente que los conceptos más visitados son *Hatha-Yoga* y *Ha*, frente a *Asana*, *Tha* y *Loto* que tienen un número de visitas bastante más bajo y próximo entre sí.

Tal y como se construye la matriz de transiciones MT_p^k, la suma de las celdas en una fila y una columna encabezadas por el mismo concepto producen un valor casi siempre idéntico. La posible diferencia entre el resultado de sumar la fila y la columna de un concepto en MT_p^k, se debe al inicio y al final de la navegación de cada usuario en la presentación EC_p^k.

Concretamente para cada usuario, la suma de la fila del concepto al que se asocia el primer ítem visitado es mayor en uno que la suma de su columna. Del mismo modo, la suma de la columna del último concepto visitado es mayor en uno que la suma de su fila. Esta mínima diferencia se acumula de usuario a usuario y se propaga desde las matrices de presentación MT_p^k a la matriz de transiciones de memorización, MT_m.

En los ejemplos utilizados se ha ignorado esta diferencia, haciendo coincidir la suma de la fila y la columna de cada concepto en las MT_p^k utilizadas. No obstante, esto no menoscaba el proceso de análisis desarrollado, ya que a menudo las diferencias de un usuario son compensadas con las de otros, y las que sobreviven en las MT_p^k son anuladas al combinarse entre sí. De modo que en la MT_m final la diferencia entre la suma de la fila y la columna de un concepto es despreciable, máxime cuando el volumen de visitas es suficiente.

3.4 Matrices de transiciones de autor (MT-autor)

Por supuesto, para poder informar al autor de las diferencias significativas entre sus modelos de representación y los modelos mentales que de éstos mantienen los usuarios es imprescindible realizar una comparación entre ambos.

Principalmente, con este objetivo, se construyen y mantienen las matrices de transiciones MT_p^k y MT_m. A través de las cuáles se captura el proceso navegacional de los usuarios estudiados, permitiendo además obtener otra información de interés para el autor como la que se describe en las secciones anteriores.

Para llevar a cabo la comparación, se requiere una estructura similar a las matrices de transiciones explicadas, pero que, en este caso, corresponda a los modelos de representación realmente disponibles en el sistema hipermedia. Con este objetivo, el Sistema de Aprendizaje construye las matrices de transición de los modelos definidos por el autor durante las fases de memorización y presentación.

Estas matrices son denominadas de autor, **MT-autor**, para distinguirlas de las matrices explicadas anteriormente. Además, a diferencia de éstas, son *matrices estáticas*, en el sentido de que no cambian con la navegación de los usuarios sino sólo cuando el autor reestructura los modelos de representación haciendo uso de la correspondiente acción evolutiva.

El sistema crea una matriz de transiciones estática para la estructura conceptual de memorización a la que denominaremos **MT_m-autor**. Esta matriz tiene las mismas dimensiones que la matriz de transiciones MT_m dinámica definida anteriormente y a la

que, a partir de ahora, notaremos **MT_m-usuarios** para hacer más claro el significado de ambas matrices y sus diferencias.

> **Def 21.5 [MT_m-autor]** La matriz de transiciones MT_m-autor es una matriz estática que recoge la existencia o ausencia de relaciones semánticas entre cada dos conceptos de la EC_M definida por el autor. El valor de una celda MT_m-autor[c_i, c_j] puede ser 1 o 0 según exista o no una relación desde c_i hasta c_j. Este valor sólo es reconsiderado cuando el autor utiliza acciones evolutivas en el Sistema de Memorización que puedan afectar al dominio conceptual de la EC_M.

Una matriz de transiciones MT_m-autor tiene *s* filas y *s* columnas, siendo *s* el número de conceptos presentes en la EC_M. El valor de cada una de las *s*×*s* celdas (i:1. .s, j:1. .s) se calcula como sigue:

- La celda MT_m-autor[c_i, c_j] es igual a 1 si existe en la EC_M una o más relaciones conceptuales que tienen como origen el concepto c_i y como destino el concepto c_j. Esto es, MT_m-autor[c_i, c_j] = 1 si $\exists\ r_c \in Rc(EC_M)$ tal que $r_c: c_i \rightarrow c_j$.
- El resto de celdas tiene valor 0. Esto es, MT_m-autor[c_i, c_j] = 0 si $r_c \notin Rc(EC_M)$ tal que $r_c: c_i \rightarrow c_j$.

Observe en la figura 5 la MT_m-autor creada para la EC_M que se muestra en el ejemplo de la figura 4.

	Loto	Ha	Asana	Hatha	Y-Y	Tha
Loto	0	0	1	0	0	0
Ha	0	0	1	1	0	0
Asana	0	0	0	1	0	0
Hatha	0	0	0	0	0	1
Y-Y	0	1	0	0	0	0
Tha	0	0	0	0	0	0

Figura 5: MT_m-autor para la EC_M de ejemplo

Si se compara esta matriz de transiciones con su respectiva matriz de transiciones dinámica, MT_m-usuarios (figura 4), es fácil reconocer algunas diferencias.

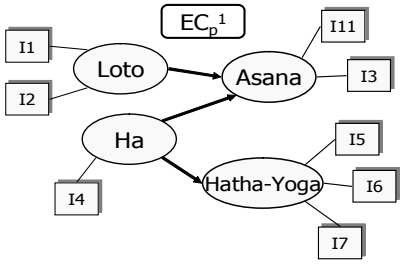
Por ejemplo, en MT_m-usuarios prácticamente todas las celdas excepto las que identifican relaciones imposibles (marca *X*), tienen un valor mayor que 0, lo que significa que algunos usuarios han seguido esa relación. Sin embargo, en la MT_m-autor únicamente hay seis celdas con valor no nulo, las cuales corresponden precisamente a las seis relaciones conceptuales que existen en la EC_M.

Del mismo modo, en la MT_m-usuarios el valor de una celda indica si la relación que representa ha sido seguida por muchos (valor alto) o pocos usuarios (valor bajo). Mientras que en la MT_m-autor todas las celdas no nulas tienen el mismo valor, 1, indicando la existencia absoluta de la relación.

Un procedimiento similar al explicado se sigue para obtener las matrices de transición estáticas asociadas a las estructuras conceptuales de presentación definidas por el autor. Notamos MT_p^k-autor a la matriz estática de una presentación EC_p^k, la cual tiene el mismo orden que la matriz dinámica MT_p^k correspondiente (a la que en adelante notaremos MT_p^k-usuarios), es decir *n*, si éste es el número de conceptos mostrados en dicha presentación.

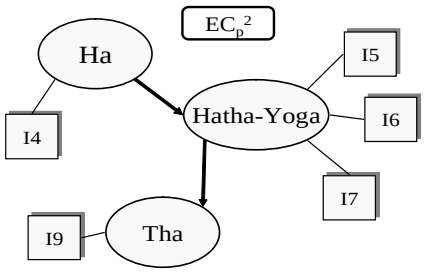
Def 21.6 [MT_p^k-autor] El valor de cada una de las $n \times n$ celdas de la matriz estática MT_p^k-autor indica la existencia o ausencia de una determinada relación conceptual en la presentación EC_P^k definida por el autor. Concretamente, la celda MT_p^k-autor[c_i, c_j] es igual a 1 si se han incluido en la EC_p^k una o más relaciones conceptuales que tienen como origen el concepto c_i y como destino el concepto c_j, en otro caso es 0.

En la figura 6 se muestran las matrices de transiciones estáticas, MT_p^1-autor y MT_p^2-autor, creadas a partir de cada una de las dos presentaciones existentes.



MT_p^1-autor

	Loto	Ha	Asana	Hatha
Loto	0	0	1	0
Ha	0	0	1	1
Asana	0	0	0	0
Hatha	0	0	0	0



MT_p^2-autor

	Ha	Hatha	Tha
Ha	0	1	0
Hatha	0	0	1
Tha	0	0	0

Figura 6: MT_p-autor para las EC_P de ejemplo

De nuevo, es inmediato advertir las diferencias existentes entre estas matrices y las correspondientes matrices dinámicas que se mostraron en la figura 4.

3.5 Transformación de las matrices de transición dinámicas (MT-usuarios)

Para comparar adecuadamente una matriz dinámica con su matriz estática correspondiente, ya sea la MT_m-usuarios con la MT_m-autor o una MT_p^k-usuarios con la MT_p^k-autor pertinente, ambas deben expresarse en los mismos términos. Esto es, es necesario eliminar o reducir las diferencias de significado que ahora mismo existen entre el contenido de ambas matrices.

Para ello, antes de hacer la citada comparación, el sistema convierte automáticamente las matrices dinámicas a matrices de ceros y unos. De manera que tras la trasformación realizada, al igual que en las matrices estáticas, los valores de las celdas de una matriz

dinámica constaten o rechacen de manera absoluta la existencia de relaciones conceptuales.

La única diferencia es que, en las matrices estáticas o de autor, la existencia es siempre real, es decir, la relación conceptual se encuentra efectivamente en la estructura conceptual de que se trate. Mientras que en las matrices dinámicas, la existencia de la relación se deriva de su utilización durante la navegación tradicional de esa estructura conceptual, y no tiene por qué corresponder en todos los casos con una existencia efectiva.

3.5.1 Transformación de MT_p^k-usuarios

El procedimiento empleado para transformar una matriz dinámica MT_p^k-usuarios[n×n] en una matriz de ceros y unos es el siguiente:

Paso 1. Para no perder la información almacenada en la matriz MT_p^k-usuarios todas las transformaciones son realizadas sobre una copia de ésta.

$\forall_{i:1..n}, \forall_{j:1..n}$ MT_p^k-usuarios'$[c_i, c_j]$ = MT_p^k-usuarios$[c_i, c_j]$

Paso 2. Se ignora la diagonal principal de la matriz, es decir la diagonal descendiente que parte de la celda MT_p^k-usuarios'$[c_1, c_1]$ y termina en la celda MT_p^k-usuarios'$[c_n, c_n]$.

Ésta recoge para cada concepto c_i la relación conceptual consigo mismo que, como se explicó con anterioridad, normalmente obedece a un deseo de obtener más información acerca del concepto y no a una relación cíclica.

Paso 2.1 Se obtiene en ***d*** la suma de las celdas situadas en la diagonal principal.

$$d = \sum_{i=1}^{n} MT_p^k\text{-usuarios'}[c_i, c_i]$$

Paso 2.2 Se descuenta *d* del número de visitas totales realizadas sobre la EC_P^k.

visitas'(EC_P^k) = visitas(EC_P^k) – d

Paso 3. Se calcula el valor medio teórico, **mt**, de una celda de la matriz de transiciones. Para ello, se divide el número de visitas totales entre el número de celdas de la matriz.

En esta división se ignora la diagonal principal, de modo que en el numerador no se contabilizan sus visitas y en el denominador se descuentan sus *n* celdas.

$$mt = \frac{visitas(EC_p^{\ k}) - d}{(n \times n) - n} = \frac{visitas'(EC_p^{\ k})}{n^2 - n}$$

Paso 4. El valor de cada celda MT_p^k-usuarios'$[c_i, c_i]$, salvo la diagonal principal, se divide entre el valor medio teórico obtenido en el paso anterior.

De este modo, el valor de la celda MT_p^k-usuarios'$[c_i, c_i]$ deja de ser el número absoluto de veces que se ha seguido la relación conceptual $r_c{:}c_i \rightarrow c_i$ para convertirse en un indicador de si ese número está por encima o por debajo de la media teórica.

$$\forall_{i:1..n}, \forall_{j:1..n} \text{ tal que } i \neq j: \quad MT_p^{\ k}\text{ - usuarios' }[c_i, c_j] = \frac{MT_p^{\ k}\text{ - usuarios' }[c_i, c_j]}{mt}$$

Paso 5. El indicador MT_p^k-usuarios'$[c_i, c_j]$ es ahora un número real comprendido en el intervalo $[0, n^2\text{-}n]$.

Este indicador tiene valor mínimo 0 cuando la relación $r_c{:}c_i \rightarrow c_j$ no se ha seguido ninguna vez (0/*mt* = 0), valor 1 cuando la relación se ha seguido el número de veces que marca la media teórica (*mt*/*mt* = 1) y valor máximo n^2-n cuando la relación es la única que se ha seguido en la presentación (v/(v/(n^2-n))).

A partir de este indicador, la transformación en 0 ó 1 del valor de cada celda MT_p^k-usuarios'$[c_i, c_j]$ se hace en función de un umbral u_{rc}:

Paso 5.a MT_p^k-usuarios'$[c_i, c_j]$ se transforma en 1 si supera o iguala el umbral u_{rc}.

$\forall_{i:1..n}, \forall_{j:1..n}$ tal que $i \neq j$ y MT_p^k-usuarios'$[c_i, c_j] \geq u_{rc} \rightarrow MT_p^k$-usuarios'$[c_i, c_j] = 1$

Paso 5.b MT_p^k-usuarios'$[c_i, c_j]$ se transforma en 0 si está por debajo del umbral u_{rc}.

$\forall_{i:1..n}, \forall_{j:1..n}$ tal que $i \neq j$ y MT_p^k-usuarios'$[c_i, c_j] < u_{rc} \rightarrow MT_p^k$-usuarios'$[c_i, c_j] = 0$

Por defecto u_{rc} vale 1, esto significa que sólo toman valor 1 las relaciones conceptuales que han sido seguidas un número de veces igual o superior al valor medio teórico.

No obstante, el autor puede modificar este valor, teniendo en cuenta que si elige un valor $u_{rc} = z$, está estableciendo que para considerare la existencia de una relación conceptual debe haber sido seguida $z \times mt$ veces (z veces el valor medio teórico).

Algoritmo 1. Transformación de MT_p^k-usuarios

La figura 7.1 muestra el resultado de ejecutar los pasos 1, 2, 3 y 4 del algoritmo que acabamos de explicar, sobre las matrices MT_p^1-usuarios y MT_p^2-usuarios tomadas del ejemplo de la figura 4.

MT_p^1-usuarios

159	Loto	Ha	Asana	Hatha
Loto	4	5	10	10
Ha	15	12	16	5
Asana	5	6	8	20
Hatha	5	25	5	8

d = 4+12+8+8 = 32
visitas'(EC_p^1) = 159-32=127
mt = 127/(4^2-4) = 127/12 ≈ 10,58

MT_p^1-usuarios'

127	Loto	Ha	Asana	Hatha
Loto	•	0,47	0,94	0,94
Ha	1,41	•	1,51	0,47
Asana	0,47	0,56	•	1,89
Hatha	0,47	2,36	0,47	•

MT_p^2-usuarios

102	Ha	Hatha	Tha
Ha	2	15	10
Hatha	5	15	20
Tha	20	10	5

d = 2+15+5 = 22
visitas'(EC_p^2) = 102-22=80
mt = 80/(3^2-3) = 80/6 ≈ 13,33

MT_p^2-usuarios'

80	Ha	Hatha	Tha
Ha	•	1,12	0,75
Hatha	0,37	•	1,5
Tha	1,5	0,75	•

Figura 7.1: Matriz de indicadores para dos MT_p^k-usuarios

Comparando la MT_p^1-usuarios inicial con la MT_p^1-usuarios' obtenida en el paso 4 del algoritmo vemos que, a parte de descartar la diagonal, tan sólo se ha hecho un cambio

de escala. Si ignoramos la diagonal y ordenamos las celdas de menor a mayor valor, veremos que el orden obtenido es idéntico para MT_p^1-usuarios y MT_p^1-usuarios'.

Por ejemplo, la relación más seguida en la EC_P^1 ha sido Hatha → Ha, concretamente en 25 ocasiones ya que MT_p^1-usuarios[Hatha, Ha] = 25. Del mismo modo, en MT_p^1-usuarios' el mayor valor es el de la celda MT_p^1-usuarios'[Hatha, Ha]. Ahora su valor es 2,36, lo que nos indica que la relación Hatha → Ha ha sido seguida en más del doble de ocasiones que en el caso de que todas las relaciones posibles en la EC_P^1 hubiesen sido navegadas el mismo número de veces (*mt*).

En la figura 7.2 se muestra el resultado final, tras ejecutar el paso 5 del procedimiento descrito. Adviértase que, ahora, las matrices MT_p^1-usuarios' y MT_p^2-usuarios' se han convertido en matrices cuadradas de ceros y unos.

127	Loto	Ha	Asana	Hatha
Loto	•	0,47	0,94	0,94
Ha	1,41	•	1,51	0,47
Asana	0,47	0,56	•	1,89
Hatha	0,47	2,36	0,47	•

u_{rc}= 1

MT_p^1-usuarios'

	Loto	Ha	Asana	Hatha
Loto	•	0	0	0
Ha	1	•	1	0
Asana	0	0	•	1
Hatha	0	1	0	•

80	Ha	Hatha	Tha
Ha	•	1,12	0,75
Hatha	0,37	•	1,5
Tha	1,5	0,75	•

u_{rc}= 1

MT_p^2-usuarios'

	Ha	Hatha	Tha
Ha	•	1	0
Hatha	0	•	1
Tha	1	0	•

Figura 7.2: Matriz de 0s y 1s para dos MT_p^k-usuarios

En el ejemplo se ha usado el umbral por defecto u_{rc} = 1, según el cual existen cuatro relaciones en la EC_P^1 (Ha → Loto, Ha → Asana, Asana → Hatha, Hatha → Ha) y tres relaciones en la EC_P^2 (Ha → Hatha, Hatha → Tha, Tha → Ha) que debido a su mayoritario seguimiento forman parte del modelo mental de los usuarios.

Si el autor hubiese elegido un valor mayor para el umbral, por ejemplo u_{rc} = 1,2, la relación conceptual Ha → Hatha habría dejado de existir en la MT_p^2-usuarios', mientras que si hubiese elegido un valor menor, quizá u_{rc} = 0,9, la existencia de las relaciones Loto → Asana y Loto → Hatha habría sido aceptada en MT_p^1-usuarios'.

3.5.2 Transformación de MT_m-usuarios

El mismo procedimiento explicado en la sección anterior, algoritmo 1, se usa para transformar en 0s y 1s las celdas de la matriz de transiciones dinámica MT_m-usuarios[s×s] asociada a la estructura conceptual de memorización.

La única diferencia es que en este caso, para calcular el valor medio teórico no se deben contabilizar las celdas marcadas con *X*. Por ello, para calcular la media teórica es

necesario descontar el número de celdas con la marca *X*, teniendo cuidado de no restar una misma celda dos veces. Esto es, si una celda marcada con *X* pertenece a la diagonal principal no se tiene otra vez en cuenta.

En la ecuación 9 se muestra la redefinición de la formula aplicada en el paso 3 del algoritmo. Como puede verse, los únicos cambios realizados, corresponden al nuevo orden de la matriz, *s*, y a que en el denominador se resta también el número de celdas cuyo valor es la marca especial *X*.

$$x = \text{número de celdas } MT_m[c_i, c_j] = X, \text{ donde i:1. .s, j:1. .s y } i \neq j \qquad (9)$$

$$mt = \frac{visitas(EC_m) - d}{((s \times s) - s) - x} = \frac{visitas'(EC_m)}{(s^2 - s) - x}$$

En la figura 8 se exhibe la MT_m-usuarios' generada a partir de la MT_m-usuarios que se muestra en la figura 4.

d =54 visitas'(EC_M) = 261-54 = 207
x =14 mt = 207/6²-6-14 = 207/16 ≈ 12,93

u_{rc}= 1

	Loto	Ha	Asana	Hatha	Y-Y	Tha
Loto	•	0,38	0,77	0,77	X	X
Ha	1,16	•	1,23	1,54	X	0,77
Asana	0,38	0,46	•	1,54	X	X
Hatha	0,38	2,3	0,38	•	X	1,54
Y-Y	X	X	X	X	•	X
Tha	X	1,54	X	0,77	X	•

MT_m-usuarios'

	Loto	Ha	Asana	Hatha	Y-Y	Tha
Loto	•	0	0	0	X	X
Ha	1	•	1	1	X	0
Asana	0	0	•	1	X	X
Hatha	0	1	0	•	X	1
Y-Y	X	X	X	X	•	X
Tha	X	1	X	0	X	•

Figura 8: Matriz de 0s y 1s para dos MT_m-usuarios

Analizando el contenido de la MT_m-usuarios' obtenida, es inmediato darse cuenta de que de la navegación de los usuarios se deducen siete relaciones conceptuales frente a las seis existentes en la EC_M real. Esto implica que los usuarios aceptan, al menos una relación que no aparece físicamente en la EC_M creada por el autor.

Además, no todas las relaciones existentes en la EC_M han sido identificadas en la MT_m-usuarios'. Este es el caso de la relación Loto → Asana presente en la EC_M y cuya celda en MT_m-usuarios' vale 0.

3.6 Análisis de diferencias entre MT-autor y MT-usuarios

La matriz de transiciones *MT-autor* representa la **estructura conceptual real** creada por el autor usando las herramientas del sistema, y la correspondiente matriz de transiciones *MT-usuarios'* representa la **estructura conceptual virtual** definida indirectamente por los usuarios a través de su navegación.

Ambas matrices se expresan en los mismos términos, esto es, 1 para poner de manifiesto la existencia de una relación conceptual y 0 para reflejar la inexistencia de la misma. Por lo tanto, el **proceso de comparación** consiste en identificar las diferencias

entre ambas matrices y puede ser realizado mediante una resta matricial elemento a elemento.

Por supuesto, la interpretación que el sistema hace de la matriz diferencia, va a depender de si el análisis se está realizando sobre una estructura conceptual de presentación EC_P^k o sobre la estructura conceptual de memorización EC_M.

3.6.1 Sugerencias sobre la EC_M

Para realizar el análisis de diferencias entre la **EC_M virtual** definida por la navegación colectiva de los usuarios, y la **EC_M real** creada por el autor, es necesario realizar la resta de las matrices de transición correspondientes, o sea MT_m-usuarios' y MT_m-autor.

Notaremos **R_m** a la matriz resultante de *restar elemento a elemento las matrices de transición MT_m-usuarios' y MT_m-autor*. La matriz R_m tiene obviamente el mismo orden que las matrices restadas.

$$R_m = MT_m\text{-usuarios'} - MT_m\text{-autor}$$

$$\text{dimensión}(R_m) = \text{dimensión}(MT_m\text{-usuarios'}) = \text{dimensión}(MT_m\text{-autor}) = s \times s$$

Por supuesto, en la matriz resta, también se ignora la diagonal principal. Esto es:

$$\forall_{i:1..s},\ R_m[c_i, c_i] = \bullet$$

$$\forall_{i:1..s}, \forall_{j:1..s} \text{ tal que } i \neq j,\ R_m[c_i, c_j] = MT_m\text{-usuarios'}[c_i, c_j] - MT_m\text{-autor}[c_i, c_j]$$

La resta de una celda marcada con *X*, siempre produce el valor *X*. Esto permite que se conserven con la marca *X* en R_m las celdas que identifican relaciones conceptuales imposibles.

$$\text{Si } MT_m\text{-usuarios'}[c_i, c_j] = X \text{ entonces } R_m[c_i, c_j] = X$$

Una vez obtenida la matriz R_m, el sistema estudia sus celdas y en base al valor que contienen, propone al autor una serie de información y modificaciones sobre la estructura conceptual de memorización.

Una celda de la matriz R_m puede contener únicamente cinco valores diferentes: *X*, •, 0, 1 y –1, donde la interpretación de cada valor es realizada por el sistema tal y como se indica en la tabla 7.

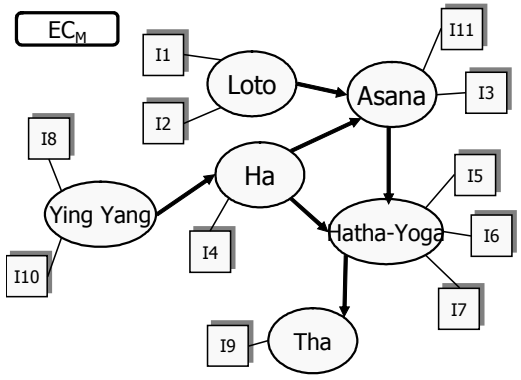
Tabla 7: Análisis de diferencias EC_M virtual – EC_M real

$R_m [c_i, c_j]$	**Interpretación y sugerencia**
•	Este símbolo identifica a una celda de la diagonal principal y por ello es ignorado en el análisis.
X	Esta marca, tal y como se indicó en la sección 3.3.2, permite detectar relaciones imposibles y conceptos olvidados. La sugerencia ante un concepto olvidado es que se elimine de la EC_M o se incluya en alguna presentación existente o futura.
1	Este valor indica que la relación conceptual r_c: $c_i \rightarrow c_j$ existe en la EC_M virtual pero no en la EC_M real. La sugerencia es **crear** una nueva relación r_c: $c_i \rightarrow c_j$ en la EC_M.

-1	Este valor indica que la relación conceptual r_c: $c_i \rightarrow c_j$ existe en la EC_M real pero no en la EC_M virtual. La sugerencia es **quitar** esa relación r_c: $c_i \rightarrow c_j$ de la EC_M.
0	Este valor indica que las celdas de ambas matrices coinciden. Lo que se traduce en que la relación r_c: $c_i \rightarrow c_j$ existe o no existe en ambas EC_M, real y virtual. Con lo cual no hay ninguna sugerencia al respecto.

Se ha realizado el proceso de análisis e interpretación explicado para la EC_M del ejemplo, utilizando para ello su MT_m-autor (figura 5) y su MT_m-usuarios' (figura 8).

Como resultado se han obtenido las sugerencias que se muestran en la figura 9. Estas son: quitar la relación Loto → Asana, y crear tres nuevas relaciones Ha → Loto, Hatha → Ha y Tha → Ha.



Sugerencias sobre la EC_M

	Loto	Ha	Asana	Hatha	Y-Y	Tha
Loto	•		quitar		X	X
Ha	crear	•			X	
Asana			•		X	X
Hatha		crear		•	X	
Y-Y	X	X	X	X	•	X
Tha	X	crear	X		X	•

Figura 9: Sugerencias al autor sobre la EC_M

Observe, que la permanencia de cada una de las relaciones conceptuales en la EC_M real es sometida a evaluación, excepto la relación que implica al concepto olvidado *Ying-Yang*, ya que es imposible obtener información sobre su aceptación o rechazo en la EC_M virtual.

De las otras cinco relaciones evaluadas, todas excepto la relación Loto → Asana se mantienen puesto que también existen en la EC_M virtual, razón por la que no aparece ninguna sugerencia al respecto.

En cualquier caso, no es aconsejable que el autor realice las modificaciones sugeridas sobre la EC_M sin estudiar también el análisis que el sistema realiza sobre cada presentación. Esto se debe a que tal y como se explica más adelante, el análisis de las presentaciones está relacionado con el análisis de la EC_M y viceversa.

Para ilustrar esta necesidad, supongamos, por ejemplo, que en una EC_M el sistema aconseja eliminar una relación conceptual escasamente utilizada. Puede ser que esto sea debido a que la relación se incluye en muy pocas presentaciones (quizá solo una). Pero que, sin embargo, allá donde se muestra es ampliamente seguida por los usuarios. En esta situación, el autor quizá considere más oportuno añadir la relación en más presentaciones o de cualquier modo mantener la relación, muy útil en las contadas presentaciones donde aparece.

3.6.2 Sugerencias sobre una EC_P^k

En la EC_M real el sistema aconseja añadir o eliminar una relación conceptual según corresponda o no con una relación en la EC_M virtual. Sin embargo, que una relación sea oportuna en la EC_M no quiere decir que también lo sea en todas sus presentaciones. Por este motivo es necesario, también, un análisis a nivel de presentación. De modo que para cada EC_P^k existente, se aconseje ocultar o mostrar relaciones conceptuales dependiendo de si éstas son innecesarias o convenientes de acuerdo a la navegación realizada por el grupo de usuarios.

Para realizar el análisis de diferencias entre una **EC_P^k virtual** y la correspondiente **EC_P^k real**, es necesario realizar la resta de las matrices de transición asociadas. En esta ocasión, notamos **R_p^k** a la matriz resultante de la operación, que al igual que las matrices implicadas tiene orden $n \times n$.

$$R_p^k = MT_p^k\text{-usuarios'} - MT_p^k\text{-autor}$$

$$\forall_{i:1..n},\ R_p^k[c_i, c_i] = \bullet$$

$$\forall_{i:1..n}, \forall_{j:1..n} \text{ tal que } i \neq j,\ R_p^k[c_i, c_j] = MT_p^k\text{-usuarios'}[c_i, c_j] - MT_p^k\text{-autor}[c_i, c_j]$$

De nuevo, el sistema estudia las celdas de la matriz diferencia y como resultado propone una serie de modificaciones en la estructura conceptual de presentación correspondiente. Ahora, cada celda de la matriz R_p^k contiene uno de los cuatro valores siguientes: •, 0, 1 y –1. Siendo la interpretación de cada valor la que se explica en la tabla 8.

Tabla 8: Análisis de diferencias EC_P^k virtual – EC_P^k real

$R_p^k[c_i, c_j]$	**Interpretación y sugerencia**
•	Este símbolo identifica a toda celda de la diagonal principal y por ello es ignorado en el análisis.
1	Este valor indica que la relación conceptual r_c: $c_i \rightarrow c_j$ aparece en la EC_P^k virtual pero no está incluida en la EC_P^k real. La sugerencia es **mostrar** una nueva relación r_c: $c_i \rightarrow c_j$ en la EC_P^k.
-1	Este valor indica que la relación conceptual r_c: $c_i \rightarrow c_j$ aparece en la EC_P^k real pero no en la EC_P^k virtual. La sugerencia es **ocultar** esa relación r_c: $c_i \rightarrow c_j$ en la EC_P^k.
0	Este valor indica que la relación r_c: $c_i \rightarrow c_j$ aparece o no aparece tanto en la EC_P^k real como en la EC_P^k virtual. Por lo tanto, no hay sugerencias al respecto.

Una vez obtenido el conjunto de sugerencias en una presentación, es conveniente cotejarlo con las obtenidas en la EC_M, con el fin de identificar situaciones como la comentada anteriormente, esto es, en la presentación se mantiene o aconseja añadir una relación conceptual mientras que en la EC_M se insta a eliminar dicha relación.

Con esta intención, en la tabla de sugerencias para una estructura conceptual de presentación EC_P^k, las etiquetas "mostrar" y "ocultar" son reforzadas con información adicional, que contrasta esa sugerencia con las sugerencias realizadas en la EC_M y el estado actual de ésta.

Concretamente, la **etiqueta "mostrar"** asociada a una relación r_c:$c_i \rightarrow c_d$ en una EC_P^k puede tener cuatro formatos distintos según la relación r_c:$c_i \rightarrow c_d$ ...

a) ... exista actualmente en la EC_M y se aconseja mantenerla (mostrar)

b) ... exista en la EC_M, pero se ha sugerido quitarla (mostrar)

c) ... no existe en la EC_M pero se ha sugerido crearla (mostrar)

d) ... no existe en la EC_M y no se sugiere su creación (mostrar).

De este modo, el formato de la etiqueta "mostrar" avisa al autor de si la sugerencia hecha sobre la relación r_c en la EC_P^k es compatible con las sugerencias realizadas sobre la EC_M. Concretamente, existe compatibilidad en los casos a) y c), mientras que las sugerencias son incompatibles en los casos b) y d).

Las sugerencias realizadas sobre r_c en la EC_P^k y en la EC_M son **compatibles** si pueden llevarse ambas a cabo. En **a)** no hay ningún problema en mantener la relación r_c en la EC_M al mismo tiempo que se muestra en la EC_P^k. En **c)** es posible llevar a cabo ambas sugerencias, creando primero la relación en la EC_M y mostrándola después en la EC_P^k.

Cuando **no existe compatibilidad** es imposible seguir ambas sugerencias, por lo que el autor se ve obligado a decidirse por una (o ninguna) de ellas. Por ejemplo, en **b)** si el autor decide eliminar la relación r_c de la EC_M ya no va a poder mostrarla en la EC_P^k. Ante esta incompatibilidad el autor debe plantearse si mantiene la relación en la EC_M para poder hacer uso de ella en la presentación EC_P^k o prescinde de ésta.

También el formato de la etiqueta "mostrar" indica si la sugerencia puede realizarse sobre la EC_P^k sin hacer nuevos cambios en la EC_M (a, b) o por el contrario necesita de alguna modificación (c, d). O lo que es lo mismo, informa de si **actualmente la relación que se sugiere mostrar existe** (**a, b**) o **no** (**c, d**) en la EC_M.

Por su parte, la **etiqueta "ocultar"** también presenta un formato diferente según la relación conceptual, r_c: $c_i \rightarrow c_d$, cuya ocultación se sugiere ...

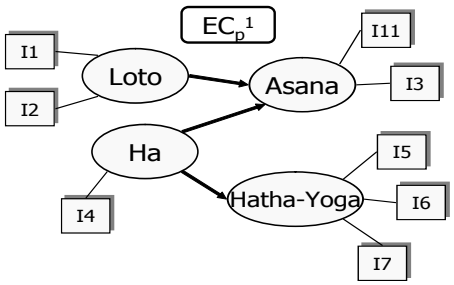
a) ... exista actualmente en la EC_M y se aconseje mantenerla (ocultar) o

b) ... exista en la EC_M, pero se ha sugerido su eliminación (ocultar).

En esta ocasión sólo hay dos alternativas, ya que si una relación no existe en la EC_M no puede mostrarse en ninguna de sus presentaciones, y por lo tanto, no cabe que el sistema sugiera su ocultación en una EC_P^k. Esto hace que en los dos casos (a y b), la modificación propuesta en EC_P^k sea **compatible** con la sugerencia realizada en la EC_M, y que no sea obligatoria la ejecución de ningún cambio en la EC_M para poder plasmar la sugerencia hecha sobre la EC_P^k.

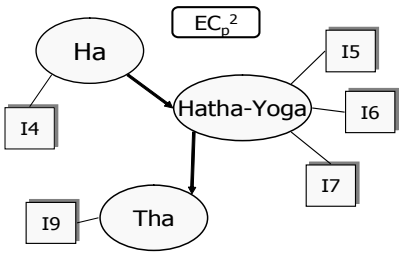
Por lo tanto, la única diferencia entre **a)** y **b)** es que en esta última, si el autor decide realizar la modificación sugerida sobre la EC_M, no tiene que realizar ya ninguna modificación en la EC_P^k, puesto que al eliminar la relación r_c en la EC_M ésta se oculta automáticamente en todas las presentaciones que la contengan a través del mecanismo de propagación del cambio.

La figura 10 visualiza las sugerencias obtenidas para las dos presentaciones, EC_P^1 y EC_P^2, cuyas matrices de transiciones, estáticas y dinámicas, se muestran en las figuras 6 y 7 respectivamente.



Sugerencias sobre EC_p^1

	Loto	Ha	Asana	Hatha
Loto	•		ocultar	
Ha	mostrar	•		ocultar
Asana			•	mostrar
Hatha		mostrar		•



Sugerencias sobre EC_p^2

	Ha	Hatha	Tha
Ha	•		
Hatha		•	
Tha	mostrar		•

Figura 10: Sugerencias al autor sobre dos EC_P

En la EC_P^2 la única sugerencia consiste en mostrar una relación semántica que vaya desde el concepto *Tha* al concepto *Ha*. El formato de la etiqueta mostrar indica al autor que esa relación actualmente no existe en la EC_M pero que su creación ha sido sugerida.

Para llevar a cabo todas las sugerencias propuestas para la EC_P^1 el autor debe crear en la EC_M las relaciones Ha→Loto y Hatha→Ha, mostrar esas relaciones y la relación Asana→Hatha en la presentación actual, ocultar en ésta la relación Ha→Hatha y decidir si elimina la relación Loto→Asana en la EC_M o sólo la oculta en la presentación actual.

4. RETROALIMENTACIÓN EN LA NAVEGACIÓN POR CONCEPTOS

Cuando el usuario recorre una presentación o la estructura conceptual de memorización completa haciendo uso del modo de navegación por conceptos, el sistema analiza su comportamiento, ya que de nuevo se trata de un proceso de navegación libre que permite extraer netamente las estrategias de navegación de los usuarios.

Obviamente, el análisis realizado a partir de la navegación por conceptos coincide con el anterior en el uso de las matrices de transiciones de autor. Es decir, las matrices de transiciones estáticas, MT_m-autor y $\forall k$ MT_p^k-autor, son construidas para capturar las relaciones conceptuales que existen en las estructuras definidas por el autor durante las fases de memorización y presentación. Una vez construidas, sólo cambian cuando lo hacen las estructuras que las generaron y se utilizan tanto en el análisis de la navegación tradicional como en el de la navegación por conceptos.

El análisis realizado durante la navegación por conceptos es idéntico al explicado para la navegación tradicional, salvo en unos cuantos aspectos, derivados casi en su totalidad de la diferente unidad de navegación empleada, el ítem en la navegación tradicional y el concepto en la navegación por conceptos.

4.1 Matrices de transiciones de presentación (MT_p^k-$usuarios_c$)

Para cada presentación EC_P^k el sistema construye e inicializa una matriz de transiciones $\mathbf{MT_p^k\text{-}usuarios_c}$ en la misma forma que se explicó para la navegación tradicional. Después el contenido de la matriz es actualizado a medida que un usuario recorre en modo conceptual esa presentación. De manera que la celda MT_p^k-$usuarios_c[c_i, c_j]$ incrementa en uno su valor cada vez que un usuario ha seleccionado el concepto c_j justo después de seleccionar el concepto c_i o lo que es lo mismo cuando el usuario ha solicitado consecutivamente los resúmenes de c_i y c_j.

A partir de la MT_p^k-$usuarios_c$ el sistema puede extraer un contador de visitas, $\mathbf{visitas_c(EC_P^k)}$, que informe del número total de conceptos seleccionados en EC_P^k. Esto permite al autor conocer el uso absoluto de esa presentación en el modo de navegación por conceptos. Véase el cálculo de $visitas_c(EC_P^k)$ en la ecuación 2' definida para la navegación conceptual a partir de la ecuación 2 de la sección 3.2.1.

$$visitas_c\left(EC_p^k\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} MT_p^k - usuarios_c\left[c_i, c_j\right] \qquad (2')$$

Asimismo, comparando el número de visitas realizadas en modo conceptual en las distintas presentaciones, el sistema obtiene un indicador relativo de visitas, $\mathbf{Ivisitas_c(EC_P^k)}$, que informa al autor de si las visitas realizadas en la presentación EC_P^k están por encima o por debajo de la media teórica. De nuevo, y como puede verse en la ecuación 3', este indicador se obtiene de la misma forma que en la navegación tradicional (sección 3.2.2).

$$Ivisitas_c\left(EC_p^k\right) = \frac{visitas_c\left(EC_p^k\right)}{\frac{\sum_{j=1}^{r} visitas_c\left(EC_p^j\right)}{r}} \text{ siendo } r \text{ el número de presentaciones} \quad (3')$$

Utilizando la información anterior, el sistema presenta al autor una lista de presentaciones ordenadas de más a menos visitas, esto es, en orden descendente según $visitas_c(EC_P^k)$. También, construye los conjuntos $\mathbf{P_c^+}$ y $\mathbf{P_c^-}$ con las presentaciones cuyo índice de visitas ($Ivisitas_c$) está por encima y por debajo de la media respectivamente.

Sin embargo, en este caso no se hace un estudio del perfil (subdominio y experiencia) de las presentaciones más o menos visitadas. Esto se debe a que si una presentación apenas se visita en modo conceptual no tiene porque significar que las características de ésta sean inadecuadas; sino, más bien, que su tamaño es suficientemente reducido, y por lo tanto, el usuario prefiere recorrerla en un modo navegación que le permita seleccionar directamente los ítems.

La lista ordenada de presentaciones y los conjuntos P_c^+ y P_c^- pueden ayudar al autor a identificar las presentaciones cuya extensión hace conveniente (o no) la navegación por conceptos. Esto es, las presentaciones incluidas en P_c^+ son presentaciones que debido a

su extensión animan a los usuarios a solicitar navegación por conceptos, con el objetivo de reducir su tamaño y evitar problemas de desorientación. Por el contrario las presentaciones en P_c^- suelen tener un tamaño reducido que hace innecesario el uso de la navegación por conceptos.

4.2 Matriz de transiciones de memorización (MT_m-usuarios$_c$)

La principal diferencia entre el análisis realizado en ambos modos de navegación libre, tradicional y por conceptos, radica en la forma en que se actualiza la matriz de transiciones dinámica de la estructura conceptual de memorización, o sea la MT_m-usuarios.

En la navegación tradicional, la MT_m-usuarios se obtiene sumando cada una de las matrices MT_p^k-usuarios obtenidas para sus presentaciones. Esto se debe a que, para reducir los problemas de desorientación, la navegación tradicional se realiza sobre una presentación y no sobre la EC_M completa. Por lo que, únicamente se puede construir el modelo mental que de la EC_M tienen los usuarios, combinando los modelos mentales que éstos poseen sobre sus distintas partes.

Sin embargo, la navegación por conceptos está pensada precisamente para recorrer desde una perspectiva global la EC_M. Por lo que, la matriz **MT_m-usuarios$_c$** se puede obtener directamente de la navegación realizada en modo conceptual sobre la EC_M. De esta manera, la celda MT_m-usuarios$_c[c_i, c_j]$ incrementa en uno su valor cada vez que un usuario ha seleccionado el concepto c_j justo después de seleccionar el concepto c_i en la EC_M.

Sobre la MT_m-usuarios$_c$ el sistema obtiene el número de visitas realizadas a conceptos de la EC_M, **visitas$_c$(EC_M)**, sumando el valor de todas sus celdas (ecuación 6'). Ahora bien, mientras que en la navegación tradicional éste coincidía con el número total de visitas (sección 3.3.1). En este caso, para obtener el número total de visitas realizadas en el modo de navegación por conceptos, **visitas$_c$**, es necesario sumar las visitas realizadas sobre cada presentación con las realizadas directamente sobre la EC_M (ecuación 10).

$$visitas_c(EC_M) = \sum_{i=1}^{s}\sum_{j=1}^{s} MT_m - usuarios_c[c_i, c_j] \neq \sum_{k=1}^{r} visitas_c(EC_p^k) \qquad (6')$$

$$visitas_c = visitas_c(EC_M) + \sum_{k=1}^{r} visitas_c(EC_p^k) \qquad (10)$$

También el cálculo del número total de visitas sobre un concepto es ligeramente distinto al explicado para la navegación tradicional (sección 3.3.3). Para obtener **visitas$_c$(c_i)** hay que sumarle a las visitas realizadas a ese concepto en cada presentación que lo incluya, las visitas realizadas sobre él en la EC_M (véase la ecuación 8').

$$\text{visitas}_c(c_i) \approx \sum_{j=1}^{s} MT_m - usuarios_c[c_i, c_j] + \sum_{k=1}^{r}\sum_{j=1}^{n} MT_p^k - usuarios_c[c_i, c_j] \qquad (8')$$

En lo que respecta a la identificación de conceptos olvidados y relaciones imposibles es indiferente si se realiza en un modo de navegación u otro, ya que no depende de la forma en que los usuarios recorren las estructuras sino de las estructuras en sí. Por lo

tanto, únicamente se realiza a partir de las matrices obtenidas en la navegación tradicional, tal y como se explica en la sección 3.3.2.

4.3 Análisis y sugerencias

De nuevo, para poder confrontar las matrices dinámicas construidas durante la navegación por conceptos con las matrices estáticas definidas por el autor, es necesario realizar un proceso de transformación que las haga comparables. Esta transformación se realiza de modo idéntico al explicado en las secciones 3.5.1 y 3.5.2 para MT_p^k-usuarios y MT_m-usuarios respectivamente.

La única diferencia en la transformación, es que para obtener MT_m-usuarios$_c$' nos despreocupamos de la marca *X* ya que debido a la forma en que ha sido construida, en MT_m-usuarios$_c$ no existen celdas marcadas con ese valor. Esto es, aunque un concepto no haya sido incluido en ninguna presentación, sí que puede ser visitado por los usuarios cuando navegan por conceptos la EC_M completa. Por este motivo, la celda de un concepto olvidado o una relación imposible puede tener, incluso un valor alto, en MT_m-usuarios$_c$.

Una vez trasformadas las matrices dinámicas, el análisis y las sugerencias realizadas para la EC_M y sus distintas presentaciones, EC_p^k, es llevado a cabo por el sistema exactamente en la misma forma en que se describió para la navegación tradicional.

En esta ocasión las relaciones identificadas como necesarias o prescindibles en la EC_M no se derivan de la navegación de sus presentaciones sino directamente de la navegación de la EC_M. Con lo cual, aunque es conveniente que el autor siga contrastando las sugerencias obtenidas en la EC_M con las de sus presentaciones, no es tan preciso como antes.

De hecho, las sugerencias realizadas sobre la EC_M tienen más peso que las realizadas sobre sus presentaciones. Debido a que la navegación conceptual está especialmente indicada para obtener una perspectiva global de la EC_M el gran volumen de visitas va a ser realizado sobre ésta, aunque puntualmente pueda utilizarse este modo de navegación también en presentaciones extensas.

Se podría pensar en combinar el análisis realizado durante la navegación tradicional y el que se lleva a cabo durante la navegación por conceptos, obteniendo así un único conjunto de sugerencias a partir de la navegación libre. Sin embargo, consideramos que es más beneficioso para el autor disponer de este doble análisis por separado. Esto le permite decidir cuál de los dos estudios prima en caso de contradicción (una relación puede ser aceptada en la navegación tradicional y rechazada en la navegación por conceptos) e identificar sugerencias dudosas cuando existe gran discordancia entre los dos modos.

Además, conceptualmente, no es muy adecuado unir las matrices de transición dinámicas que se han obtenido en los dos modos de navegación estudiados. Las relaciones capturadas durante la navegación tradicional de una estructura difieren de las obtenidas durante la navegación por conceptos de ésta. Esta diferencia radica en que durante la navegación tradicional, el usuario pasa de un ítem a otro sin pensar demasiado en los conceptos que está relacionando en esa transición. Mientras que en la navegación por conceptos, la transición que realiza el usuario es conceptual, esto es,

pasa de un concepto a otro, y por lo tanto, establece de un modo mucho más consciente la relación entre ellos.

En definitiva, las relaciones identificadas en las matrices MT-usuarios son definidas implícitamente por los usuarios cuando visitan ítems en modo tradicional, mientras que las relaciones en las matrices MT-usuarios$_c$ representan sus transiciones explícitas entre conceptos, con lo cual no es conveniente mezclarlas.

5. El Papel del Autor en el Proceso de Retroalimentación

A lo largo del presente capítulo se ha descrito con detalle la información registrada por el sistema durante la navegación libre que los usuarios realizan sobre las estructuras de navegación proporcionadas. Esta información se representa y almacena en las denominadas matrices de transiciones, las cuales mantienen una cuenta del número de veces que los usuarios han pasado de un concepto a otro.

A partir de la información almacenada en las matrices de transiciones, y comparando éstas con las matrices obtenidas para las estructuras definidas por el autor, el sistema realiza un proceso de análisis, tras el cual es capaz de deducir las inferencias que se resumen en la tabla 9.

Estas inferencias se pueden dividir en tres grupos dependiendo de cuál sea su principal función:

a) Informar al autor del **uso** que los usuarios hacen de las estructuras de navegación. El objetivo es permitir al autor distinguir las presentaciones o conceptos prescindibles de aquellos ampliamente utilizados por los usuarios. También permite comparar el uso del sistema en modo tradicional y por conceptos.

b) Hacer notar al autor ciertas **características** de sus estructuras, como por ejemplo, los subdominios de las presentaciones más visitadas, la existencia de conceptos no incluidos en ninguna presentación, el rango de experiencia que abarca una presentación poco utilizada, etc. El objetivo es que el autor modifique las estructuras, reforzando las características positivas y reparando las negativas.

c) Orientar al autor sobre las **diferencias** existentes entre las estructuras que el definió y las estrategias de navegación llevadas a cabo sobre éstas por los usuarios. El objetivo que se persigue es hacer converger las estructuras de navegación hacia los modelos mentales que los usuarios tienen de éstas.

Tabla 9: Información proporcionada al autor

	Tradicional	**Por conceptos**
Uso de las estructuras definidas por el autor (a)		
Número total de visitas en el sistema	×	×
Número total de visitas en cada presentación	×	×
Número total de visitas en la EC_M		×
Número total de visitas a cada concepto	×	×

Listado de presentaciones ordenado de mayor a menor número de visitas	×	×
Características de las estructuras definidas por el autor (b)		
Conjunto P^+ con las presentaciones más visitadas	×	×
Conjunto de subdominios capturados en todas las presentaciones de P^+	×	
Conjunto de subdominios capturados en alguna presentación de P^+	×	
Conjunto P^- con las presentaciones menos visitadas	×	×
Intervalo de experiencia unión para las estructuras de aprendizaje definidas sobre cada presentación en P^-	×	
Intervalo de experiencia intersección para las estructuras de aprendizaje definidas sobre cada presentación en P^-	×	
Intervalo de experiencia medio para las estructuras de aprendizaje definidas sobre cada presentación en P^-	×	
Conceptos no incluidos en ninguna presentación (conceptos olvidados)	×	
Pareja de conceptos no incluida en ninguna presentación (relaciones imposibles)	×	
Convergencia de las estructuras de autor y los modelos mentales de los usuarios (c)		
Relaciones conceptuales que deben ocultarse en una presentación	×	×
Relaciones conceptuales que deben eliminarse en la EC_M	×	×
Relaciones conceptuales que deben mostrarse en una presentación	×	×
Relaciones conceptuales que deben crearse en la EC_M	×	×

En definitiva, el objetivo de todas estas sugerencias es dotar al autor de información útil y significativa para que éste pueda aproximar sus modelos de representación a los modelos mentales comunes a la mayoría de los "usuarios libres" de su sistema.

Sin embargo, como ya se comentó antes, es el autor quien tiene la decisión final de seguir las sugerencias obtenidas o no, todas o sólo parte. Tomada la decisión de aceptar una sugerencia, el autor lleva a cabo la modificación o modificaciones oportunas mediante la ejecución de un determinado conjunto de acciones evolutivas, lo que garantiza la consistencia tras el cambio.

La tabla 10 recoge las acciones evolutivas que con más frecuencia van a ser ejecutadas para hacer efectivas las sugerencias realizadas. Las acciones incluidas son aquellas que afectan a las relaciones conceptuales, tanto en el Sistema de Memorización, como en el de Presentación [García, 01c], así como las que permiten modificar el etiquetado de una estructura en el Sistema de Aprendizaje (capítulo 14).

Tabla 10: Evolución tras las sugerencias

Crea-asociación-conceptual (c_o, c_d, r_c, EC_M)	Crea r_c:$c_o \rightarrow c_d$ en la EC_M
Borra-asociación-conceptual(<c_o, r_c, c_d>, EC_M)	Elimina r_c:$c_o \rightarrow c_d$ en la EC_M y el concepto c_d si queda desconectado

Modifica-asociación-conceptual(c_n, <c_o, r_c, c_d>, EC_M)	Elimina r_c:$c_o \rightarrow c_d$ y crea r_c:$c_o \rightarrow c_n$ ó r_c:$c_n \rightarrow c_d$, en la EC_M
Muestra-AC (<c_o, r_c, c_d>, EC_P^k)	Muestra r_c:$c_o \rightarrow c_d$ en la EC_P^k
Oculta-AC (<c_o, r_c, c_d>, EC_P^k)	Oculta r_c:$c_o \rightarrow c_d$ en la EC_P^k
Acciones evolutivas para modificar el etiquetado de una EC_P^k: AñadirSubdominioEC_P, EliminarSubdominioEC_P, ModificarSubdominioEC_P.	
Acciones evolutivas para modificar el etiquetado de una EC_A^j: ModificarExperienciaEC_A	

Otra cuestión por determinar es ¿cuándo realiza el sistema el análisis descrito? Y, ¿cuándo proporciona el resultado de dicho análisis al autor? Respecto a estas cuestiones, consideramos que las sugerencias deben ser proporcionadas al autor cuando éste lo requiera. Y para su comodidad planteamos la posibilidad de programar una generación automática de sugerencias cada *x* visitas realizadas en el sistema.

Cabe remarcar que el número total de visitas debe ser significativo para que los análisis realizados tengan sentido, aunque de nuevo esto es, en última instancia, responsabilidad del autor. En cualquier caso, el autor debe ser consciente de que las inferencias obtenidas a partir del análisis tradicional son fiables en cuanto que *visitas(EC_M)* sea suficientemente grande, del mismo modo que para poder confiar en el análisis de la navegación por conceptos el valor de *$visitas_c$* debe ser elevado.

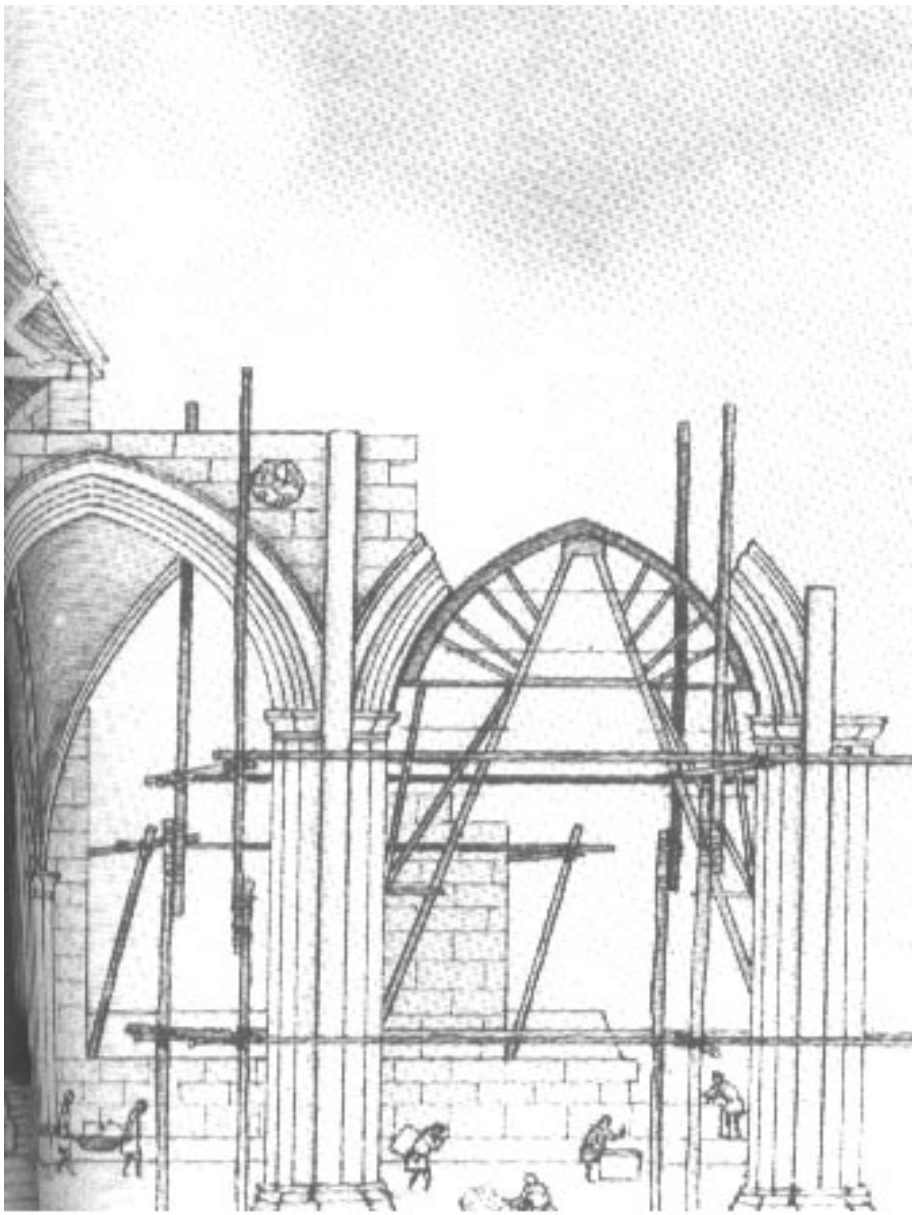
Una vez que se proporcionan las sugerencias, sean seguidas o no, el autor puede solicitar la puesta a cero de las matrices de transiciones para excluir la información ya analizada en los próximos análisis. En caso de no hacerlo, las visitas realizadas por los futuros usuarios se suman sobre las visitas recogidas actualmente en las correspondientes matrices de transición.

Es recomendable que después de realizar cambios significativos en las estructuras, el autor ponga a cero las matrices de transición para separar el uso de éstas antes y después de la transformación sufrida. Por lo tanto, cuando el autor lleva a cabo las sugerencias propuestas por el sistema es conveniente que reinicialice las matrices.

Por supuesto, para comprobar el grado real de eficacia de los mecanismos de inferencia propuestos y la utilidad para el autor de las sugerencias e información proporcionada, es necesario realizar un amplio estudio experimental que permita evaluar la técnica de retroalimentación adaptativa (capítulo 25, Conclusiones y Trabajo Futuro).

Módulo V

Implementación

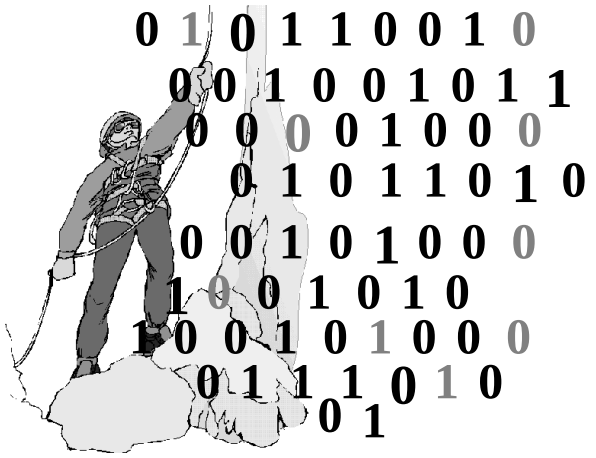


Los Pilares de la Tierra

Módulo V

CAPÍTULO 22

Implementación



Resumen

En este capítulo se presenta un prototipo que implementa parte importante de la estructura y funcionalidad formulada en el modelo SEM-HP. Se describen los requisitos que debe cumplir, así como el modelo de ciclo de vida llevado a cabo durante su construcción. Se justifican y explican las principales decisiones de diseño que se han tomado, y se hace una especificación del prototipo utilizando UML. Por último se muestra la herramienta en acción durante la creación y navegación de un pequeño sistema hipermedia adaptativo cuyo dominio semántico es el paradigma orientado a objetos.

Tabla de contenidos

Implementación

1. PRESENTACIÓN DE JSEM-HP

Basado en el modelo SEM-HP se ha desarrollado un prototipo implementado en Java al que hemos llamado **JSEM-HP**. El desarrollo del prototipo ha sido realizado dentro del proyecto de investigación MEIGAS (TIC2000-1673-C06-04), el cual es un subproyecto del proyecto DOLMEN (TIC2000-1673-C06).

2. ESPECIFICACIÓN DE REQUISITOS

Antes de entrar en detalles de diseño e implementación es necesario definir, con la adecuada precisión, esto es, sin ambigüedad, las propiedades o restricciones que debe satisfacer el producto software que se quiere desarrollar. Durante esta especificación de requisitos es conveniente distinguir entre requerimientos funcionales y no funcionales [Pressman, 87][Sommerville, 88].

Los **requerimientos funcionales** se refieren a la naturaleza del funcionamiento del sistema software y se especifican en la sección 2.1. Los **requerimientos no funcionales**, descritos en la sección 2.2, expresan ciertas características que debe poseer el producto software, no acerca de qué debe hacer, sino de cómo debe hacerlo, por lo tanto, imponen restricciones en el espacio de soluciones posibles.

2.1 Requisitos funcionales

Los requerimientos funcionales pueden expresarse de forma resumida en los siguientes términos: *Se desea implementar una herramienta de autor capaz de desarrollar sistemas hipermedia adaptativos de acuerdo a lo especificado en el modelo SEM-HP*.

Así pues, el comportamiento que debe presentar la herramienta de autor puede ser separado, al igual que en el modelo, en cuatro fases: memorización, presentación, navegación y aprendizaje.

En la **fase de memorización** se espera que la herramienta ponga en manos del autor todos los instrumentos necesarios para crear una estructura conceptual de memorización. Debe ser capaz de almacenar las estructuras creadas y recuperarlas posteriormente, así como de permitir su correcta evolución. Todo esto se traduce en la necesidad de funciones destinadas a *definir y modificar redes semánticas*. Estas funciones deben permitir operaciones como las siguientes: crear un nodo (ítem o concepto), crear una asociación (funcional o conceptual), establecer propiedades para un nodo o una asociación, modificar estas propiedades, borrar un nodo, borrar una asociación, etc.

En la **fase de presentación** se espera que la herramienta permita al autor definir una o más presentaciones a partir de una estructura conceptual de memorización. Las presentaciones deben ser almacenadas, manteniendo una correspondencia con la estructura de memorización de la que derivan, para permitir posteriormente recuperar sólo las presentaciones de la estructura de memorización activa. Para crear una presentación son necesarias funciones que permitan *ocultar nodos y asociaciones en*

una red semántica. Para admitir la evolución de las presentaciones también se requieren funciones que muestren los nodos o asociaciones ocultados previamente.

En la **fase de navegación**, la herramienta debe ser capaz de definir distintas navegaciones para una misma presentación, gestionando adecuadamente su almacenamiento y recuperación. La función principal de esta fase es la creación, según lo establecido en el modelo, del *conjunto de reglas de orden* asociadas por defecto a la presentación actual. Para hacer evolucionar este conjunto de reglas deben existir funciones destinadas a: extender la navegabilidad de una relación conceptual, reducirla nuevamente, modificar la formula de autor en una regla de orden, desactivar una regla y reactivarla.

En la **fase de aprendizaje**, la herramienta debe permitir la creación y evolución de un conjunto de *reglas de actualización y conocimiento*. De nuevo, debe ser posible establecer distintos aprendizajes para una misma estructura conceptual de navegación, incluyendo los mecanismos necesarios para su adecuada recuperación. Las funciones principales en esta fase son: crear el conjunto de reglas de actualización por defecto, modificar las reglas de actualización, y añadir, borrar o modificar reglas de conocimiento.

También se requieren funciones que permitan al usuario **recorrer el sistema** desarrollado en los cuatro modos de navegación propuestos. Estas funciones pueden dividirse en tres grupos:

1. Funciones para *monitorizar la actividad del usuario*, por ejemplo saber qué ítem ha seleccionado.

2. Funciones para crear, almacenar y recuperar el *modelo del usuario*, así como para consultarlo y actualizarlo a medida que éste navega. Por ejemplo, funciones para determinar si un ítem es accesible, establecer si tras la visita de un ítem se requiere actualización, ejecutar la regla de actualización del ítem visitado, etc.

3. Funciones para implementar las *técnicas de adaptación* propias de cada modo: elección personalizada de la estructura de navegación, ocultación, deshabilitación y anotación de ítems, rutas guiadas, etc.

2.2 Requisitos no funcionales

Respecto a los requisitos no funcionales cabe destacar los **requerimientos de interfaz**. En este sentido se desea una interfaz de *ventanas*, que facilite la selección de opciones a través de menús. Además de un *menú fijo* con todas las opciones posibles en un momento dado, se requieren *menús contextuales* que sólo aparezcan al seleccionar o situarnos encima de un elemento concreto, y que muestren únicamente las opciones disponibles para éste.

La interfaz debe ser funcional y fácil de usar. Presentando, siempre que se pueda, una lista con las opciones válidas para evitar que el autor cometa errores sintácticos en la construcción o modificación de los elementos del sistema. Fundamentalmente en las reglas de orden, actualización o conocimiento que presentan una sintaxis más complicada.

También se quiere que, en todo momento, la interfaz de autor muestre la estructura conceptual de memorización, de presentación, de navegación y de aprendizaje en la que se está trabajando. Proporcionando un modo fácil de elegir otra de las estructuras existentes en cada fase, por ejemplo mediante listas desplegables.

Respecto al **sistema operativo**, se desea que la herramienta de autor, así como los sistemas hipermedia desarrollados con ésta, puedan ejecutarse con la mayor independencia posible, siendo fundamental su funcionamiento en *Windows y Linux*.

La **interacción con otros productos software** se centra en el navegador utilizado para visualizar el contenido de los ítems de información seleccionados durante la navegación del usuario. En este sentido se requiere que la herramienta *permita la utilización de distintos navegadores*: Internet Explorer, Netscape, Mozilla, etc.

Los **requerimientos de desempeño** son más o menos los de cualquier sistema de este tipo en cuanto a capacidad de memoria, rendimiento, fiabilidad, tolerancia frente a fallos, integridad y protección de datos. No se trata de un sistema con requisitos especiales de seguridad o con limitaciones en el tiempo de respuesta. Únicamente durante la generación de las rutas guiadas es necesario ajustar el consumo de espacio y tiempo.

Los **requerimientos del ciclo de vida** pueden dividirse en dos tipos:

a) Referentes al proceso de desarrollo.

b) Referentes al producto.

Respecto al **proceso de desarrollo** (a) no existen expresamente restricciones temporales, restricciones para el uso de recursos, ni obligación de ajustarse a un estándar de calidad o desarrollo concreto. Sin embargo, se prefiere el uso de *software libre* y estándares abiertos para desarrollar la herramienta.

Los requerimientos referentes al **producto** (b) son mantenibilidad, flexibilidad, portabilidad y reusabilidad. Más allá de la mantenibilidad del sistema se requiere la *capacidad evolutiva* de éste. Es decir, el sistema debe estar dotado de los mecanismos evolutivos necesarios para que el autor pueda hacer los cambios que desee en un modo flexible e íntegro, potenciando así la reusabilidad del mismo.

Respecto a los **requerimientos de operación**, se espera un nivel de preparación mínimo en el caso de los autores que van a utilizar la herramienta de autor, y nulo para los usuarios que van a navegar los sistemas hipermedia desarrollados con ésta.

3. MODELO DE CICLO DE VIDA

Se está siguiendo un **modelo de ciclo de vida en espiral** [Boehm, 88] orientado a detectar fuentes de riesgo y prevenir posibles fracasos. Las fases del modelo son fundamentalmente cuatro:

1. Planificación de objetivos, alternativas y restricciones.

2. Análisis de riesgos: Se evalúan las alternativas. Se identifican las fuentes de riesgo y se proponen y comienzan a aplicar estrategias para resolverlas.

3. Ingeniería: Se profundiza en la resolución de los riesgos pendientes y se avanza en la obtención del producto una vez eliminados. Se desarrolla el producto de siguiente nivel (construcción de prototipos).

4. Evaluación del producto: La evaluación del prototipo permite identificar nuevos requisitos o perfiles además de los ya existentes. Dependiendo del resultado de la evaluación se decide si dar por finalizado el ciclo o continuar. En este caso se planifican las fases siguientes.

En el paso 3, el término **prototipo** puede referirse a:

- Un prototipo en papel o un modelo basado en PC que describe la interacción hombre-maquina.

- Un prototipo que implemente algunos subconjuntos de la función requerida para el programa deseado.

- Un programa existente que ejecuta parte o toda la función deseada, pero que tiene algunas características que deben ser mejoradas en el nuevo trabajo de desarrollo.

En nuestro caso el prototipo desarrollado corresponde a la segunda acepción, se trata además de un **prototipo cíclico**, o sea incremental, no desechable (figura 1). Es una herramienta que implementa las funciones principales del modelo, y que en cada iteración se hace más completo y estable. Por lo que, cabe esperar, que tras pocas iteraciones llegue a convertirse en el producto definitivo.

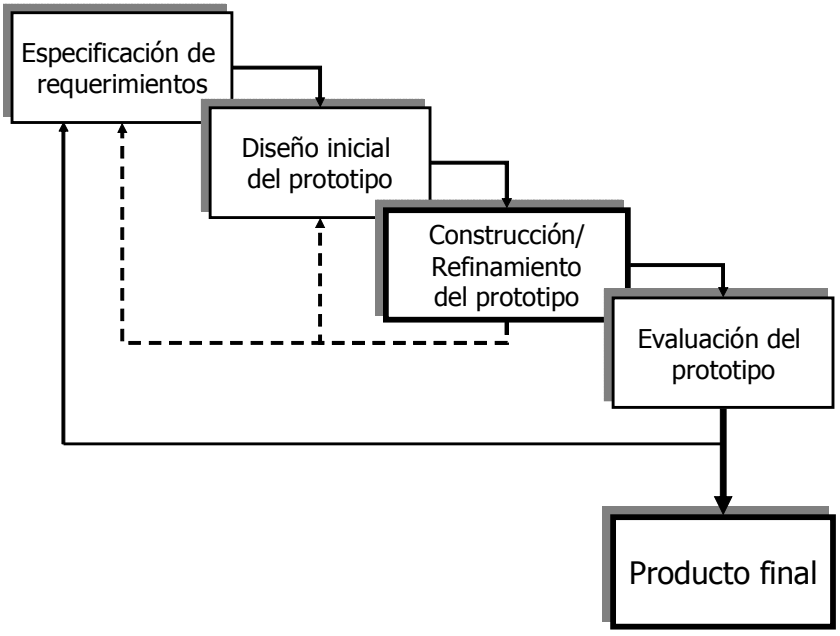


Figura 1: Desarrollo de prototipos

4. DISEÑO E IMPLEMENTACIÓN

4.1 Principales decisiones de diseño

4.1.1 El entorno de programación

El lenguaje de programación elegido para desarrollar el prototipo ha sido **Java**. Las razones que han motivado esta decisión son principalmente tres:

1) El carácter *multiplataforma* de Java,

2) su integración con la *web*, y

3) la disponibilidad de múltiples *librerías*.

Concretamente, para implementar el editor de estructuras conceptuales, nos hemos basado en una librería de grafos implementada en Java y denominada *JGraph* [Alder, 01]. Esta librería es libre y de código abierto, se encuentra en una estado maduro de desarrollo y respeta el patrón modelo vista controlador (MVC). A pesar de todas estas características favorables, el diseño del prototipo se ha realizado intentando que sea lo más independiente posible de la librería.

El entorno de programación utilizado ha sido ***Netbeans*** *Integrated Development Enviroment* [Netbeans, 00], con *Java Development Kit* [Java, 94].

4.1.2 Dominio semántico

Durante la elaboración de la herramienta ha surgido el concepto de **dominio semántico**, publicado en [Molina, 02]. Un dominio semántico se refiere a un área de conocimiento y se define a través de los atributos de sus nodos y el conjunto de relaciones propias del área.

Por ejemplo, a pesar de su similitud fonética, en áreas de conocimiento tan diferentes como la Astronomía y la Gastronomía cabe esperar que el conjunto de relaciones conceptuales sea muy distinto, casi disjunto si exceptuamos las relaciones estándar *isA*, *akindOf* y *partOf*.

La existencia explícita de los dominios semánticos permite *reutilizar la definición de un dominio* en todos los sistemas hipermedia cuyo dominio de conocimiento pertenezca al área. Esto permite que varias estructuras conceptuales de memorización se asocien al mismo dominio semántico, y por lo tanto, utilicen las mismas relaciones conceptuales y funcionales sin necesidad de redefinirlas.

Además, en todas las estructuras creadas bajo un mismo dominio, una etiqueta semántica tiene el mismo significado. Esta propiedad garantiza una coherencia semántica entre los distintos sistemas hipermedia creados dentro de un área de conocimiento común.

4.1.3 Atributos genéricos

Para aumentar la versatilidad de la herramienta se han utilizado **atributos genéricos** [Molina, 02]. Es decir, *el número y el tipo de los atributos de un elemento del modelo no va a ser fijo, sino variable*. Esta propiedad es muy interesante desde el punto de vista de los conceptos e ítems de información, ya que sus propiedades pueden variar de un dominio semántico a otro.

Para hacer realmente flexible el uso de los atributos genéricos, es necesario incluir acciones evolutivas que permitan al autor definir los atributos que tiene cada nodo y las propiedades de cada atributo. De esta forma, el autor puede personalizar los datos que se almacenan sobre cada nodo con muy poco esfuerzo.

Puesto que los atributos de un nodo pueden cambiar, *el formulario de edición del nodo se genera en tiempo de ejecución*. Todos los atributos extienden la clase *CSAttribute* mediante la que se pueden definir sus propiedades. Para cada atributo se obteene su correspondiente *CSAttributeGUI* que se encarga de interaccionar con el autor dentro del formulario de edición del nodo. De este modo, añadir un nuevo atributo se reduce a crear un *CSAttribute* y *CSAttributeGUI* a partir de los existentes.

Por ejemplo, para los atributos de tipo *String* se definen las clases *StringAttribute* y *StringAttributeGUI*.

4.1.4 Acciones evolutivas y restricciones como clases

Para representar las **acciones evolutivas** (ACe) se ha creado una clase para cada tipo de acción, siendo un objeto de esa clase una instanciación de la acción con unos determinados parámetros. El conjunto de parámetros formales depende del tipo de acción evolutiva, y el conjunto de parámetros reales de la modificación concreta que el autor desea llevar a cabo.

Por ejemplo, para añadir un nuevo concepto en la estructura conceptual de memorización se crea una instancia de la clase *AddConcept*. Dicha clase representa la acción evolutiva deseada y sólo necesita un parámetro para identificar el nuevo concepto: su etiqueta semántica.

Después del proceso de instanciación, se envía un mensaje al meta-sistema con la acción creada y el elemento a modificar (en el ejemplo anterior, la estructura conceptual de memorización actual). Aunque para realizar la acción evolutiva se llama a un método del meta-sistema, realmente el código que se ejecuta se encuentra definido en la propia acción evolutiva.

También las **restricciones** de las acciones evolutivas se implementan como clases. La clase de una restricción contiene el código necesario para comprobarla. De este modo, cada acción evolutiva tiene una lista con las restricciones que hay que comprobar antes (prerrestricciones) y después de su ejecución (posrestricciones).

Cuando una posrestricción no se cumple, es necesario deshacer el cambio y volver al sistema a su estado original. Esto complica la ejecución de la acción evolutiva, pero hay

restricciones que es muy difícil o no se pueden evaluar sin ejecutar la acción. Por ello, se ha implementado un mecanismo de modificación orientado a **transacciones**.

El proceso de ejecutar la acción evolutiva, consistente en comprobar las restricciones y rechazar la acción si éstas no se cumplen, lo efectúa un método (*runOn*) de la clase abstracta *EvAction*. Puesto que todos los tipos de acción evolutiva heredan de esta clase, sólo tienen que definir sus parámetros, inicializar su conjunto de restricciones e implementar el método que realmente realiza los cambios (*actualRunOn*).

Se podrían haber implementado las acciones evolutivas como métodos del meta-sistema, sin embargo, al hacerlo como clases se facilita la propagación del cambio al ser posible enviar la acción, con todos sus parámetros y su resultado a los sistemas donde debe ser propagada. Además, no hay que modificar el meta-sistema para contemplar un nuevo tipo de acción evolutiva, y sería muy fácil llevar una historia estructural del sistema en caso de ser necesario (simplemente almacenando las ACe).

El hecho de tener las restricciones como clases nos proporciona también algunas ventajas, como por ejemplo la posibilidad de que varias acciones evolutivas compartan la misma restricción sin necesidad de redefinirla. Por ejemplo, comprobar que los atributos definidos como únicos no se repiten es común a las acciones *AddConcept* y *AddItem*.

4.2 Diagramas de clases

A continuación se incluye un conjunto de diagramas de clases que pretende ilustrar el diseño realizado para la implementación del prototipo. Por cuestiones de espacio, y para facilitar la comprensión de los diagramas, sólo se incluyen las clases más importantes, dejando a un lado clases auxiliares y de interfaz de usuario.

4.2.1 Estructuras conceptuales

El diagrama de la figura 2 muestra las interfaces y clases principales del prototipo JSEM-HP, atendiendo a la creación y navegación de las estructuras conceptuales.

La interfaz *AttributeValuePairs* permite definir un mapeo entre atributos y valores. Ejemplos de parejas (atributo, valor) son: el nombre de una estructura conceptual, el subdominio de conocimiento de la estructura conceptual de presentación, el identificador de un ítem, el nombre de un concepto, etc.

La interfaz gráfica es capaz de generar, en tiempo de ejecución, un editor a medida para los atributos que tiene un objeto en un determinado momento. Siempre y cuando, el objeto implemente la interfaz *AttributeValuePairs* y sus atributos sean de la clase *CSAttributes*.

La interfaz *EvolutionarySystem* representa las características que consideramos necesarias para que un modelo pueda evolucionar en SEM-HP. Entre otras, se requiere un acceso orientado a transacciones, de forma que si una acción evolutiva falla (sus poscondiciones no se cumplen) se pueda devolver el modelo a su anterior estado consistente. Todas las clases que implementan *EvolutionarySystem* definen los métodos *beginTransaction*, *commitTransaction* y *rollBackTransaction*.

La clase *MetaSystem* hace evolucionar a los sistemas (*EvolutionarySystems*) ejecutando acciones evolutivas sobre ellos. Un meta-sistema puede dirigir la evolución de varios sistemas, pero un sistema tiene asignado un único meta-sistema responsable de su evolución.

La interface *ConceptualStructure* representa las características básicas, comunes a todas las estructuras conceptuales de SEM-HP, ya sean de memorización, presentación, navegación o aprendizaje. La mayoría de estas características hacen referencia a la necesidad de atributos para describir la estructura conceptual y a la capacidad evolutiva de ésta, por lo tanto, son heredadas de las interfaces *AttributeValuePairs* y *EvolutionarySystem*.

La clase *SemanticDomain* define un área de conocimiento, descrita a través de las relaciones conceptuales posibles y sus propiedades, los roles de las asociaciones funcionales, los atributos de los elementos de la estructura conceptual (ítems, conceptos, y relaciones) y de la estructura conceptual propiamente dicha.

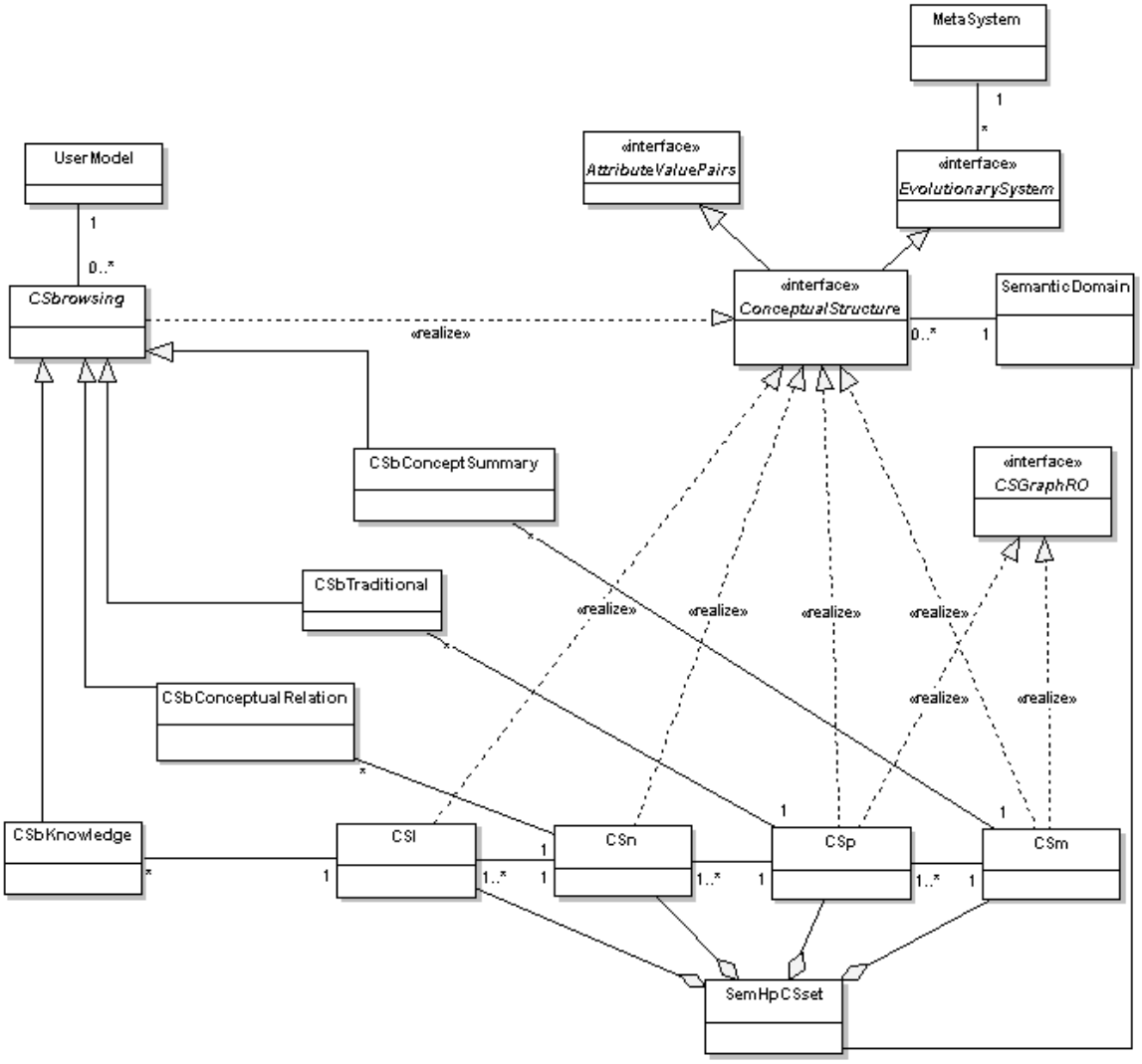


Figura 2: Estructuras conceptuales

La interfaz *CSGraphRO* define operaciones de acceso (sólo lectura) al grafo que representa la red semántica en una estructura conceptual de memorización o de presentación. Tiene operaciones para saber los nodos existentes, qué nodos están conectados con cuáles y a través de qué aristas, etc.

Las clases *CSm*, *CSp*, *CSn* y *CSl* representan respectivamente a la estructura conceptual de memorización, de presentación, de navegación y de aprendizaje. Todas implementan las operaciones básicas incluidas en la interfaz *ConceptualStructure*, y las dos primeras implementan además la interfaz *CSGraphRO* para gestionar los nodos y aristas de la red semántica que representan.

La clase *SemHpCSset* representa un conjunto de estructuras conceptuales relacionadas entre sí, todas pertenecientes a un mismo dominio semántico. Se trata de un árbol de estructuras conceptuales (ver figura 3), cuyo nodo raíz es una *CSm*, de la que cuelgan todas las *CSp* creadas a partir de ella, cada *CSp* es, a su vez, raíz de un subárbol cuyo primer nivel está formado por las *CSn* definidas sobre esa *CSp*, etc.

Esta clase tiene métodos para responder a preguntas del tipo: para esta *CSm* ¿qué *CSp* existen?, para esta *CSn* ¿qué *CSl* se han definido?, etc.

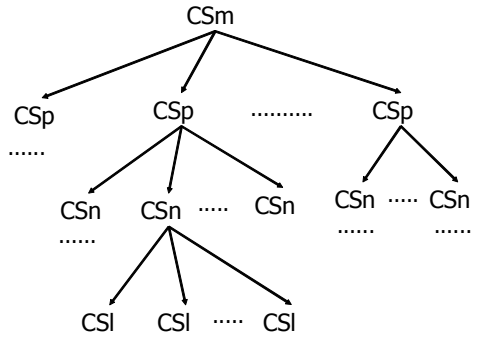


Figura 3: SemHpCSset

CSbrowsing es una clase abstracta que representa una estructura conceptual que está siendo navegada por un usuario. Se crea en el momento que el usuario selecciona un modo de navegación sobre la estructura conceptual del tipo adecuado. Contiene métodos para actualizar el modelo de usuario a medida que el usuario va navegando.

Las clases *CSbConceptSummary*, *CSbTraditional*, *CSbConceptualRelation* y *CSbKnowledge* extienden *CSbrowsing* dependiendo del modo de navegación que representan. Redefinen los métodos para comprobar si un ítem es accesible y se relacionan con una CS diferente en cada caso. Actualmente sólo están implementados *CSbTraditional* y *CSbKnowledge*. *CSbConceptualRelation* está implementada parcialmente.

La clase *UserModel* representa el modelo de usuario y contiene métodos para su inspección y actualización.

4.2.2 Estructura conceptual de memorización

El siguiente diagrama (figura 4) se centra en la clase *CSm*, y muestra sus principales relaciones de composición, herencia y asociación con otras clases.

DefaultGraphModel es la representación de un grafo en la librería JGraph, y se compone de varios elementos de la clase *DefaultGraphCell*.

DefaultGraphCell es un elemento de un grafo, nodo o arista, implementado en JGraph.

DefaultEdge es la implementación de JGraph de una arista de un grafo. Puesto que es un elemento del grafo hereda de *DefaultGraphCell*.

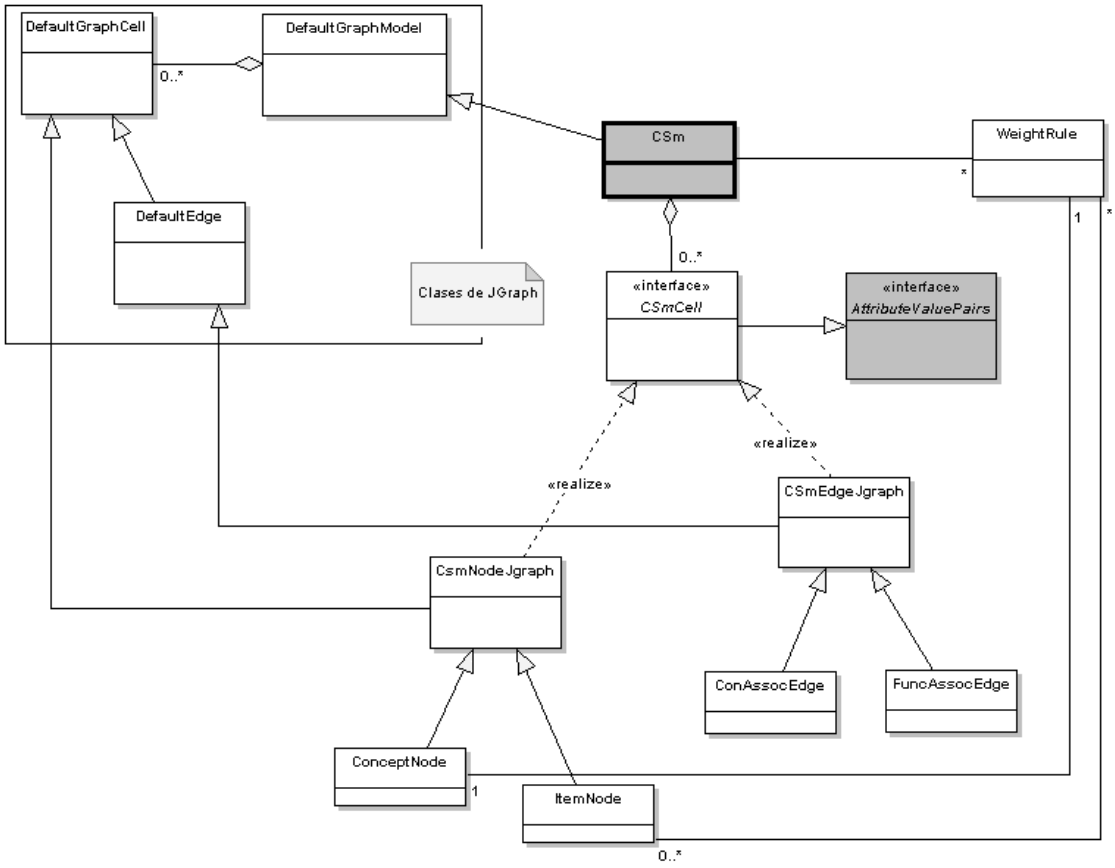


Figura 4: Contexto de *CSm*

CSmCell es una interfaz que define las operaciones comunes a cualquier elemento de una estructura conceptual de memorización (*CSm*). Hereda de la interfaz *AttributeValuePairs* porque todo elemento (concepto, ítem, asociación conceptual, asociación funcional) tienen asociado un conjunto de parejas (atributo, valor) que lo describe.

CSmEdgeJgraph y *CSmNodeJgraph* son clases que implementan la interfaz *CSmCell* y que representan respectivamente una arista y un nodo de una *CSm*. Han sido implementadas utilizando la librería JGraph.

Las clases *ConceptNode* e *ItemNode* heredan de *CSmNodeJgraph* y representan respectivamente un concepto y un ítem en una *CSm*.

Las clases *ConAssocEdge* y *FuncAssocEdge* heredan de *CSmEdgeJgraph* y representan respectivamente una asociación conceptual y funcional en una *CSm*.

4.2.3 Estructura conceptual de presentación

El diagrama de la figura 5 muestra el contexto en el que se encuentra inmersa una estructura conceptual de presentación.

CSp está implementada de forma que se almacenan los elementos ocultos (*CSmCell*) de la *CSm*, y se redefinen los métodos de *CSGraphRO* para que el grafo resultante en la *CSp* no contenga los ítems, conceptos y asociaciones ocultadas por el autor en la estructura conceptual de memorización.

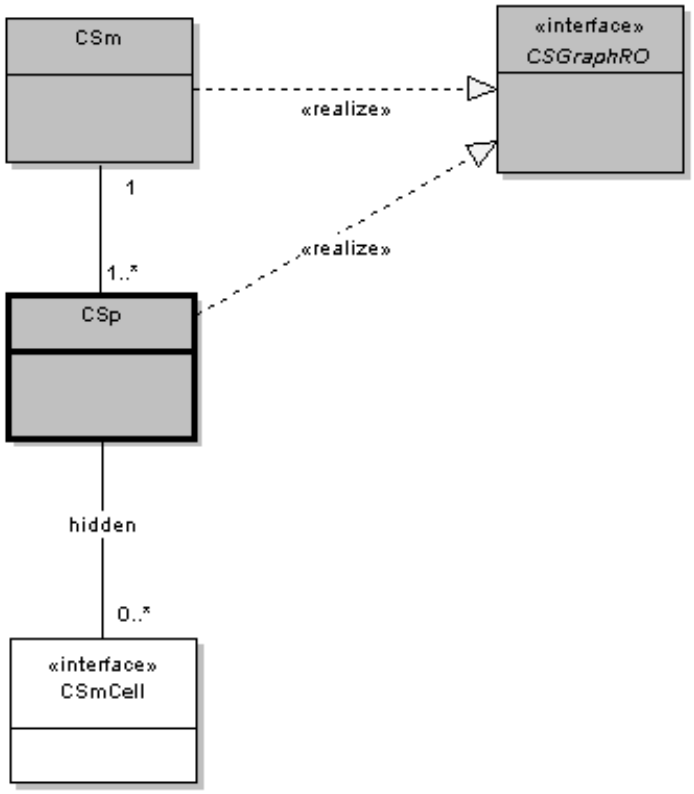


Figura 5: Contexto de *CSp*

4.2.4 Estructura conceptual de navegación

El siguiente diagrama (figura 6) se centra en la *CSn*, mostrando su relación con la *CSp* y sus principales elementos.

La clase *CSn* define un conjunto de reglas de orden (*OrderRule*) sobre una *CSp*.

La clase *OrderRule* representa una regla de orden compuesta por una cabeza y un cuerpo.

La cabeza es el ítem para el que se define la regla y pertenece a la clase *ItemNode*.

El cuerpo de la regla pertenece a la clase *OrderRuleBodyLE* que es un tipo de expresión lógica (hereda de *ComplexLogicExpression*) donde se definen los requisitos de orden para la visita del ítem cabeza.

Dicha expresión lógica se divide a su vez en dos formulas lógicas, *authorFormula* y *systemFormula*, unidas mediante el operador lógico *and*. La primera contiene predicados lógicos de la clase *Before* y la segunda de la clase *Previous*.

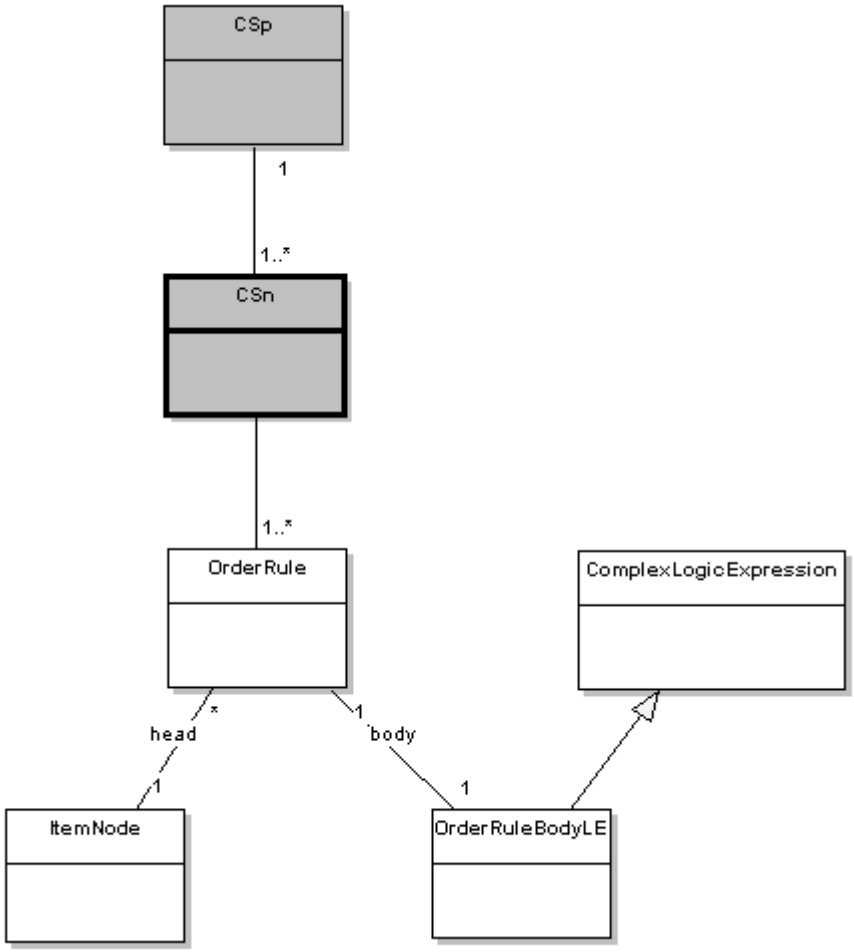


Figura 6: Contexto de *CSn*

4.2.5 Estructura conceptual de aprendizaje

El diagrama que se muestra en la figura 7 tiene como objeto mostrar los principales elementos y relaciones en torno a una *CSl*.

La *CSl* se define a partir de una *CSn* y tiene asociado un conjunto de reglas de actualización y otro de reglas de conocimiento. El primer conjunto contiene tantas reglas como ítems distintos existen en *CSn*. El segundo conjunto es inicialmente vacío y crece o decrece según el autor añada o elimine reglas de conocimiento.

La clase *KnowledgeRule* representa una regla de conocimiento definida por una cabeza y un cuerpo. La cabeza es el ítem (*ItemNode*) para cuya visita se imponen condiciones en la regla. El cuerpo es una expresión lógica de la clase *KnowledgeRuleBodyLE*.

La clase *UpdateRule* representa una regla de actualización compuesta, también, por una cabeza y un cuerpo. La cabeza es el ítem a cuya visita se asocia la regla. Y el cuerpo es una expresión lógica de la clase *UpdateRuleBodyLE*.

Tanto *KnowledgeRuleBodyLE* como *UpdateRuleBodyLE* heredan de la clase *ComplexLogicExpresion*. Sin embargo, los predicados lógicos usados en uno y otro son muy distintos. En el primero se trata de predicados binarios clásicos (*Equals*, *LargerThan*, *SmallerOrEqualThan*,...) que permiten exigir un determinado nivel conocimiento sobre un ítem. En el segundo son predicados de actualización (*IncAbs*, *FixAbs*, *IncRel*,...) definidos especialmente para actualizar el grado de conocimiento sobre un ítem.

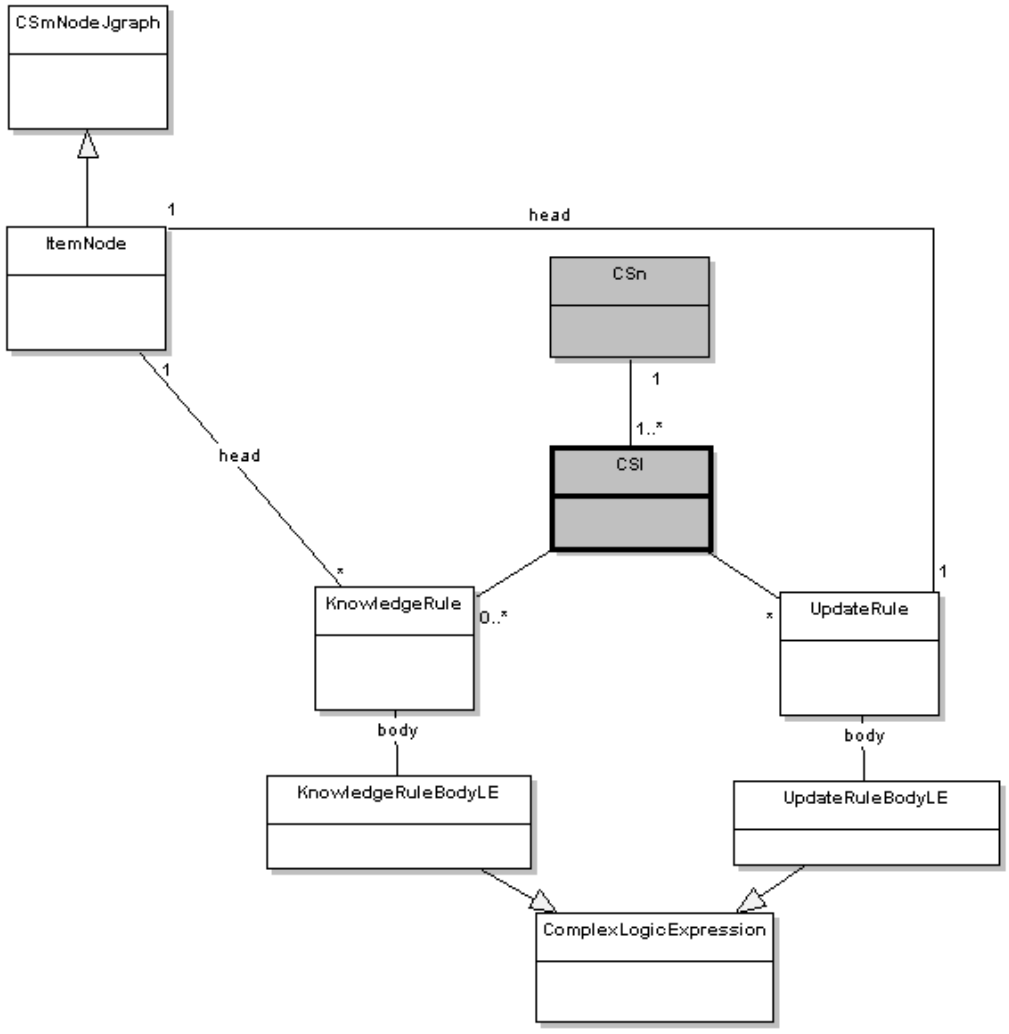


Figura 7: Contexto de *CSl*

4.2.6 Expresiones lógicas

En el siguiente diagrama (figura 8) se muestra la estructura utilizada en el prototipo JSEM-HP para construir y modificar expresiones lógicas complejas. Se desea hacer notar que la misma estructura se ha aprovechado para definir las reglas de orden, de actualización y conocimiento.

Se ha desarrollado una interfaz gráfica capaz de generar un editor interactivo para la creación y modificación de formulas lógicas con la estructura que hemos definido. Siendo fácil de particularizar para los casos de reglas de conocimiento, actualización y orden, mediante la creación de subclases de la clase abstracta *LogicSemanticDomain* que permite definir los operadores lógicos y predicados disponibles, los parámetros que pueden tener los predicados, etc.

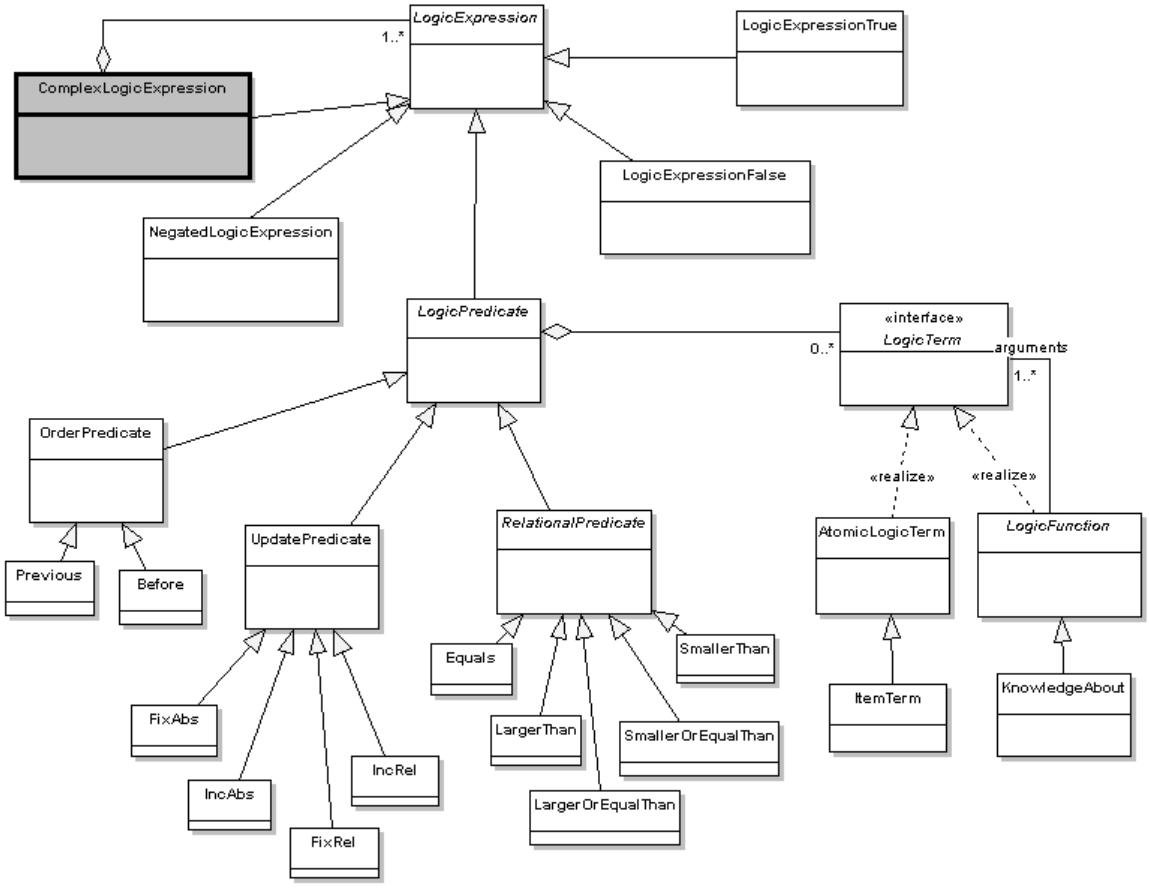


Figura 8: Expresiones lógicas

Una expresión lógica compleja (*ComplexLogicExpression*) se compone recursivamente de una o más expresiones lógicas.

Una expresión lógica (*LogicExpression*) puede ser la constante de verdad *true* (*LogicExpressionTrue*), la constante *false* (*LogicExpressionFalse),* un predicado lógico (*LogicPredicate*), una expresión compleja (*ComplexLogicExpression)* o una expresión lógica negada (*NegatedLogicExpression*).

Un predicado lógico se compone de cero o más términos lógicos sobre los que se evalúa. Un término lógico (interfaz *LogicTerm*) puede ser implementado por un término

atómico o por una función lógica. Un término atómico (*AtomicLogicTerm*) es un valor numérico, un *String* o un ítem (*ItemTerm*). Una función lógica (*LogicFunction*) requiere como argumentos uno o más términos lógicos, que pueden ser atómicos o nuevamente funciones lógicas. Un tipo de función lógica es la función de conocimiento *KnowledgeAbout*, que aplicada sobre un ítem devuelve el grado con que se conoce, esto es $K(i_j)$.

En nuestro problema necesitamos tres tipos de predicados lógicos: *OrderPredicate*, *UpdatePredicate* y *RelationalPredicate*, utilizados respectivamente en el cuerpo de las reglas de orden, de actualización y de conocimiento.

4.2.7 CSmDefault, CSpDefault, CSnDefault y CSlDefault

Aunque en los diagramas anteriores se han utilizado *CSm*, *CSp*, *CSn* y *CSl* como clases, se trata en realidad de interfaces, implementadas respectivamente por las clases *CSmDefault*, *CSpDefault*, *CSnDefault* y *CSlDefault*.

Este doble nivel ha sido ocultado para simplificar los diagramas. Sin embargo, es necesario si se quiere separar los métodos intrínsecos de la estructura conceptual, del conjunto de métodos auxiliares utilizados para implementar dichos métodos.

A continuación, para cada caso, se muestran los métodos definidos en la interfaz (CS) y los métodos implementados en la clase (CSDefault) que realiza ésta y otras interfaces (figura 2).

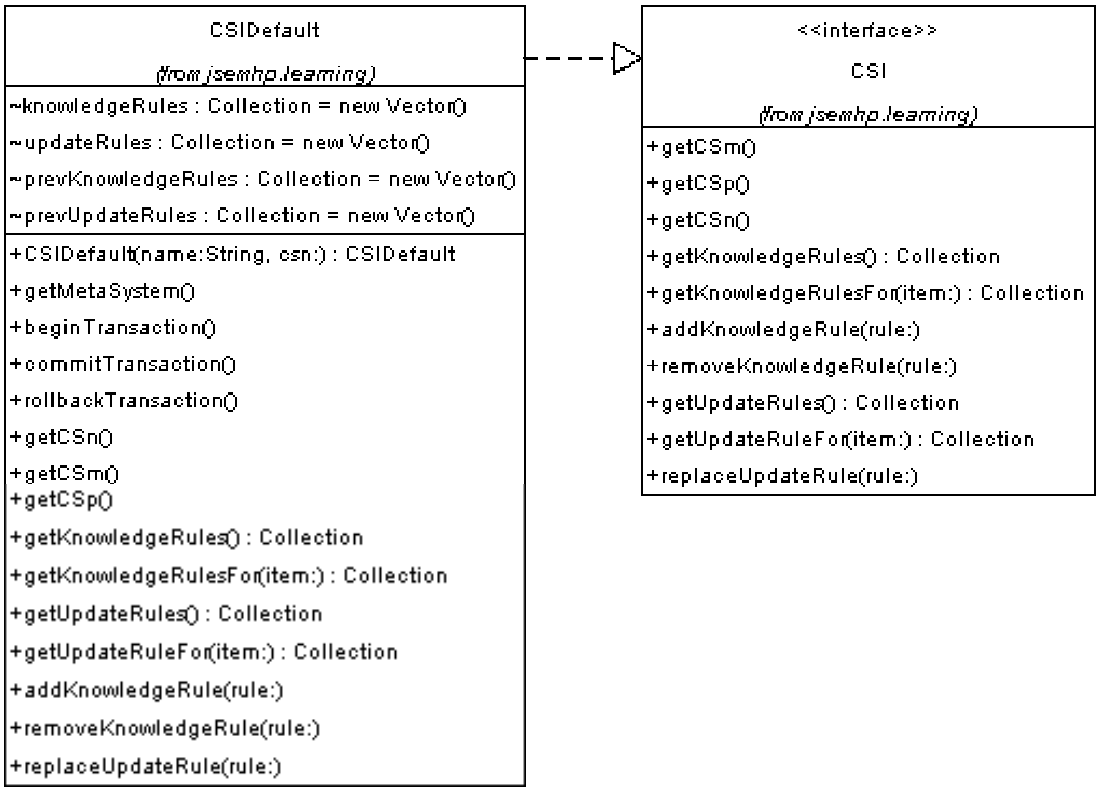


Figura 9: Métodos de *CSl*

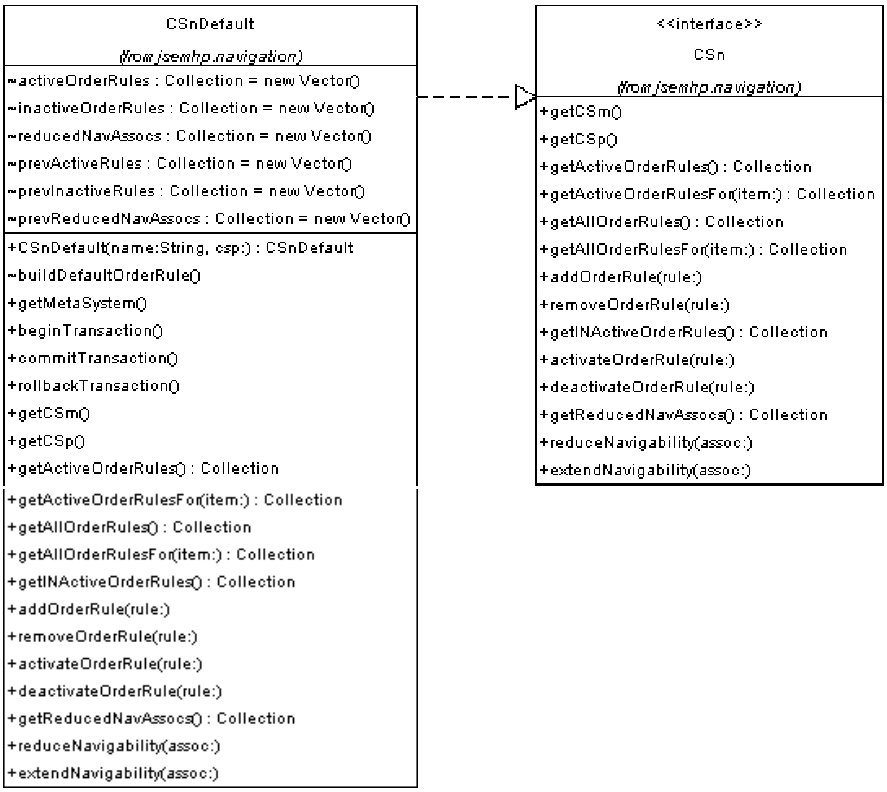


Figura 10: Métodos de *CSn*

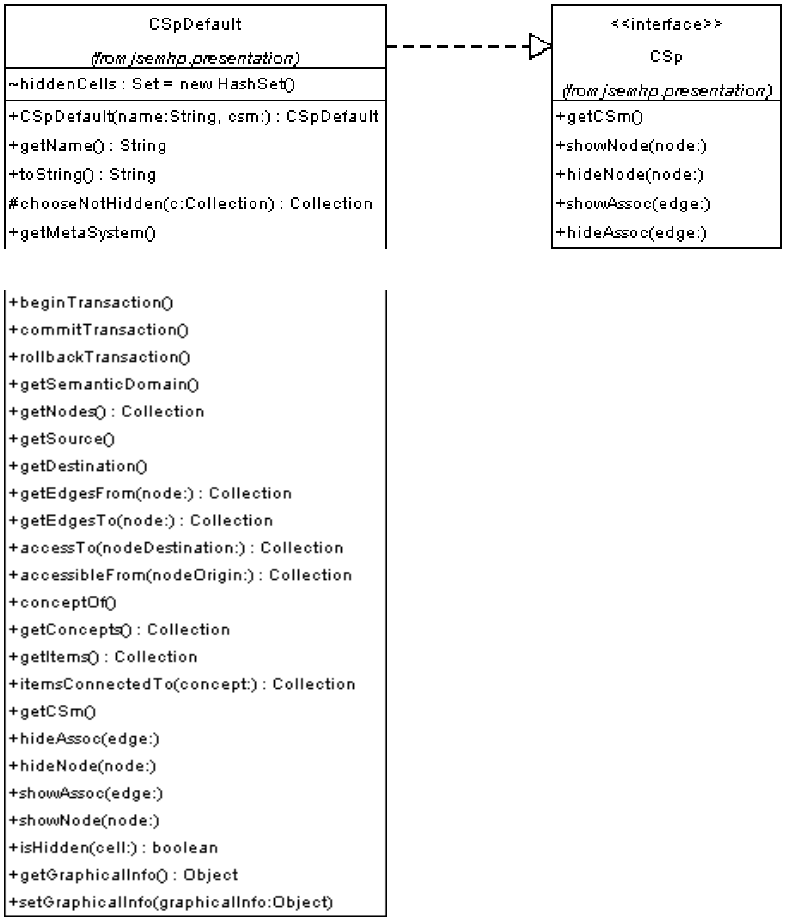


Figura 11: Métodos de *CSp*

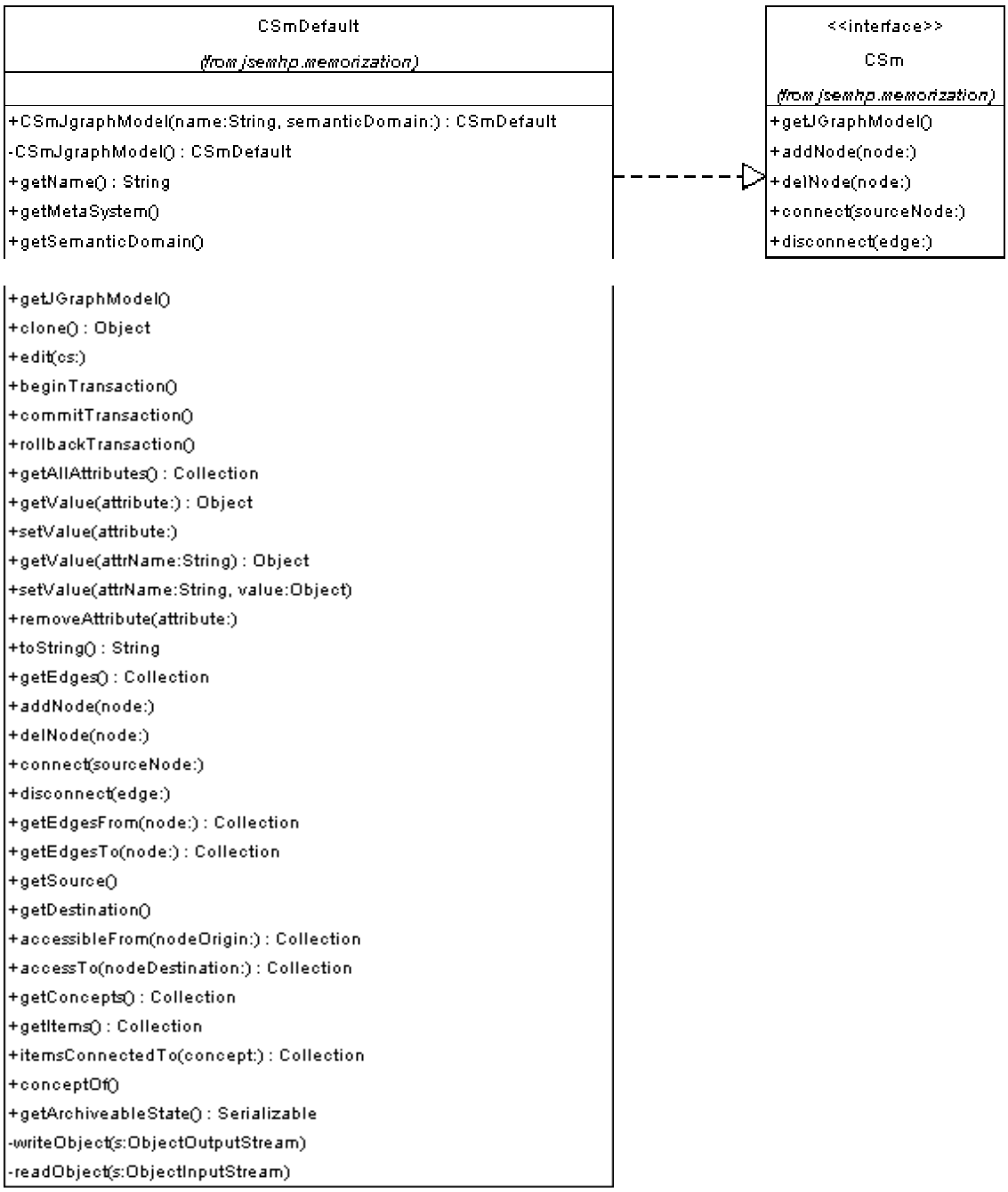


Figura 12: Métodos de *CSm*

4.2.8 Navegación por conocimiento

En el siguiente diagrama se muestran las principales clases que intervienen en la navegación por conocimiento, sus relaciones y sus métodos.

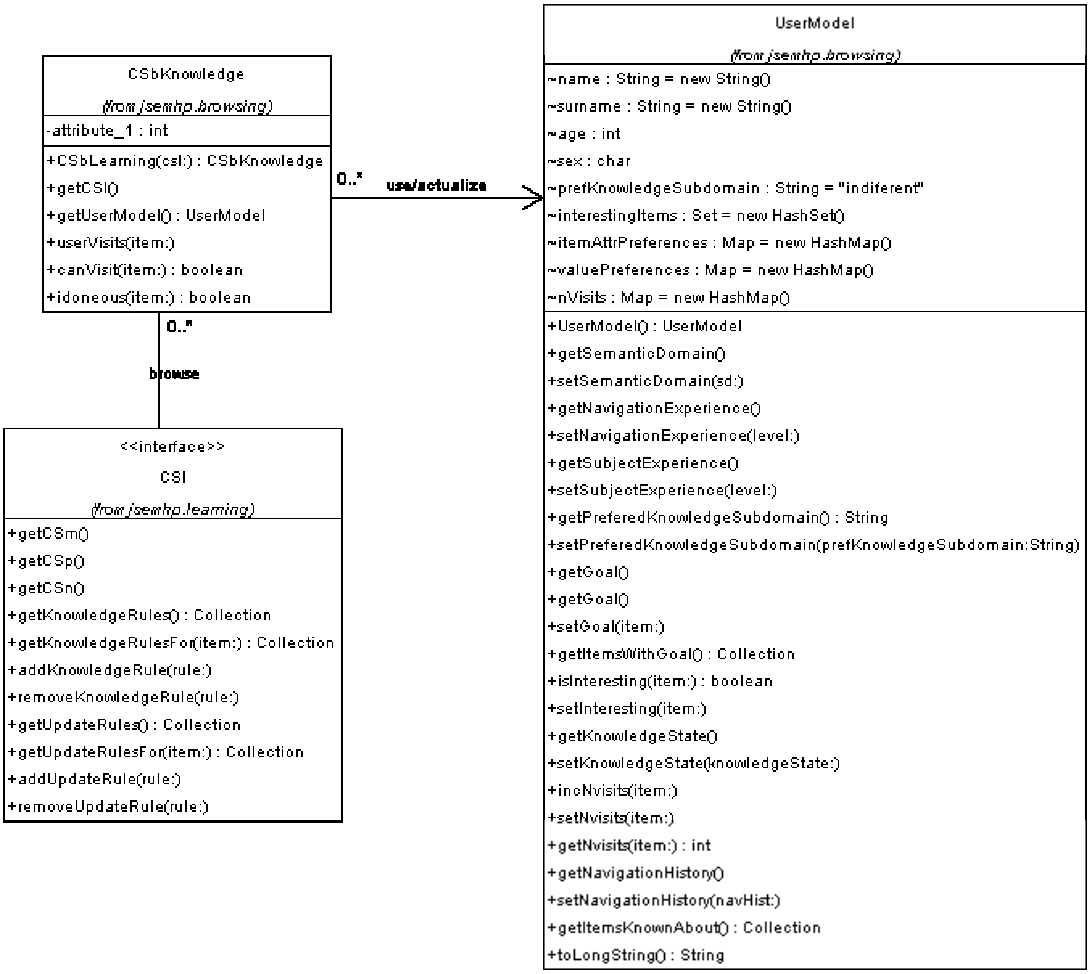


Figura 13: Navegación de una *CSl*

4.2.9 Evolución del modelo

En el patrón de diseño MVC tradicional, la vista y el controlador constituyen la interfaz del usuario para modificar el modelo, de forma que las acciones del controlador cambian directamente el modelo.

En nuestro caso, el meta-sistema es el encargado de modificar el modelo, por lo tanto el controlador debe redirigirle los cambios solicitados por el autor, a través de las acciones evolutivas correspondientes. Una vez que el modelo ha cambiado, notifica a sus vistas dicho cambio para que éstas se actualicen, tal y como se define en el patrón MVC (véase la figura 14).

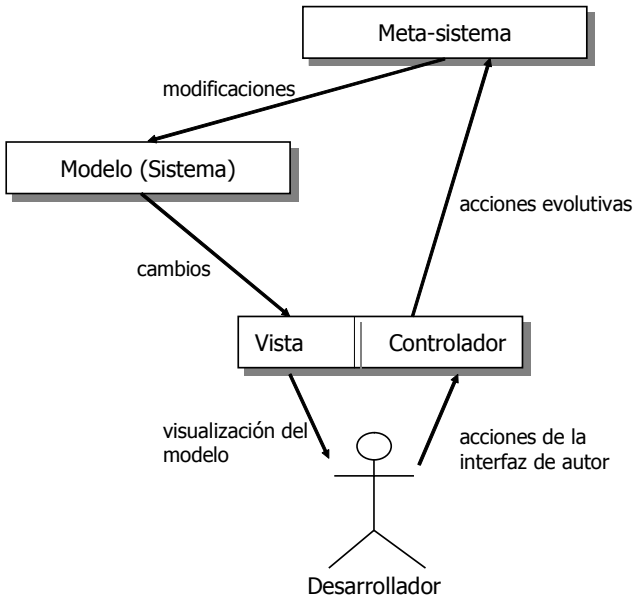


Figura 14: Modelo vista-controlador

En la figura 15 se muestran los métodos del meta-sistema (clase *MetaSystem*) y el sistema (interfaz *EvolutionarySystem*). Observe que el primero contiene métodos para dirigir la ejecución de la acción evolutiva que le envía el controlador, y el segundo para comenzar una transacción y deshacerla en caso de ser necesario.

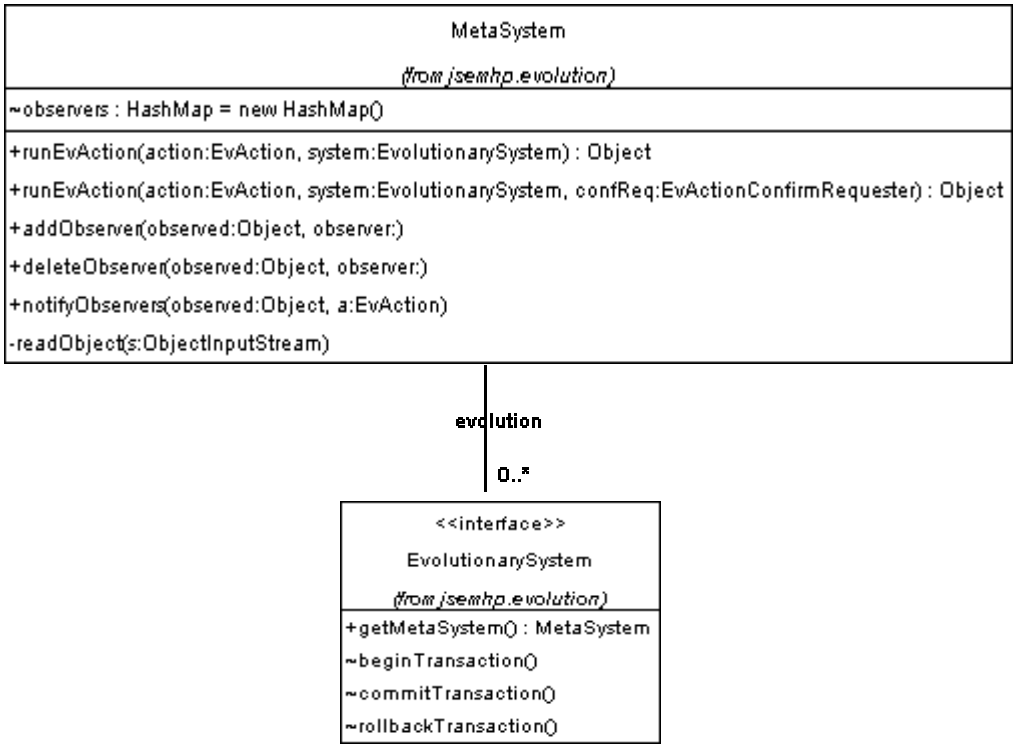


Figura 15: Sistema y Meta-Sistema

Durante la ejecución de una acción evolutiva deben comprobarse un conjunto de restricciones que ésta tiene asociadas. Los métodos de las clases abstractas *EvAction* y *Restriction* se muestran en la figura 16.

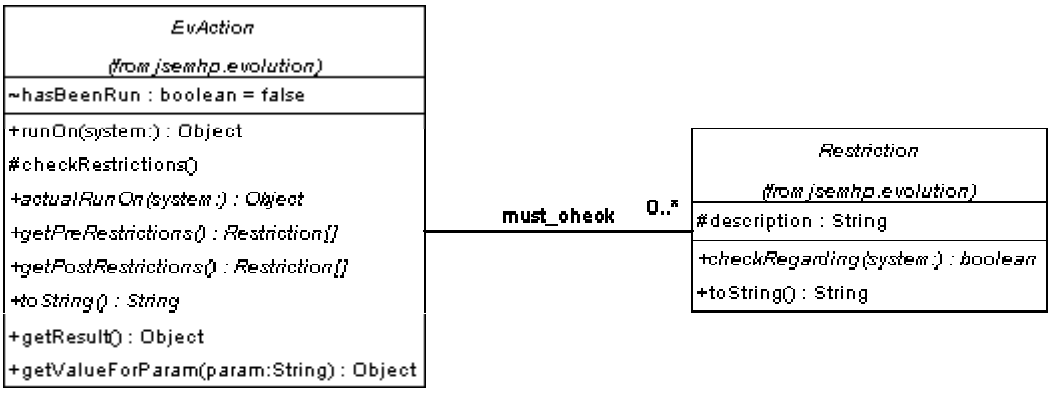
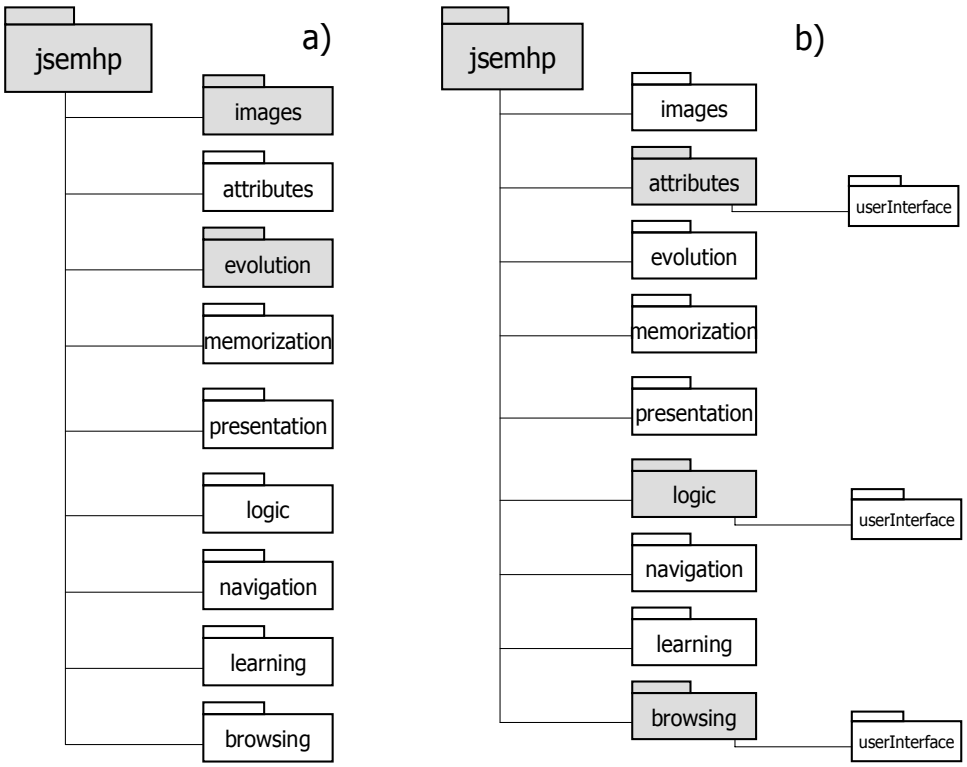


Figura 16: Acciones evolutivas y restricciones

Los métodos abstractos son implementados en las clases que extienden *EvAction* y *Restriction*. Ejemplos concretos de acciones evolutivas son: *AddItem* (añadir un ítem), *ChangeConAssoc* (modificar una asociación conceptual), *ShowFuncAssoc* (mostrar una asociación funcional), *HideConcept* (ocultar un concepto), *ChangeLogicOp* (cambiar un operador lógico), *DeactivateOrderRule* (desactivar una regla de orden), *AddKnowledgeRule* (añadir una regla de conocimiento), *RemplaceUpdateRule* (modificar una regla de actualización), etc.

4.3 Estructura de paquetes

La forma en que se organizan las clases e interfaces del prototipo corresponde al esquema de paquetes que se muestra a continuación. En cada caso, se han sombreado en gris los paquetes que se encuentran abiertos.



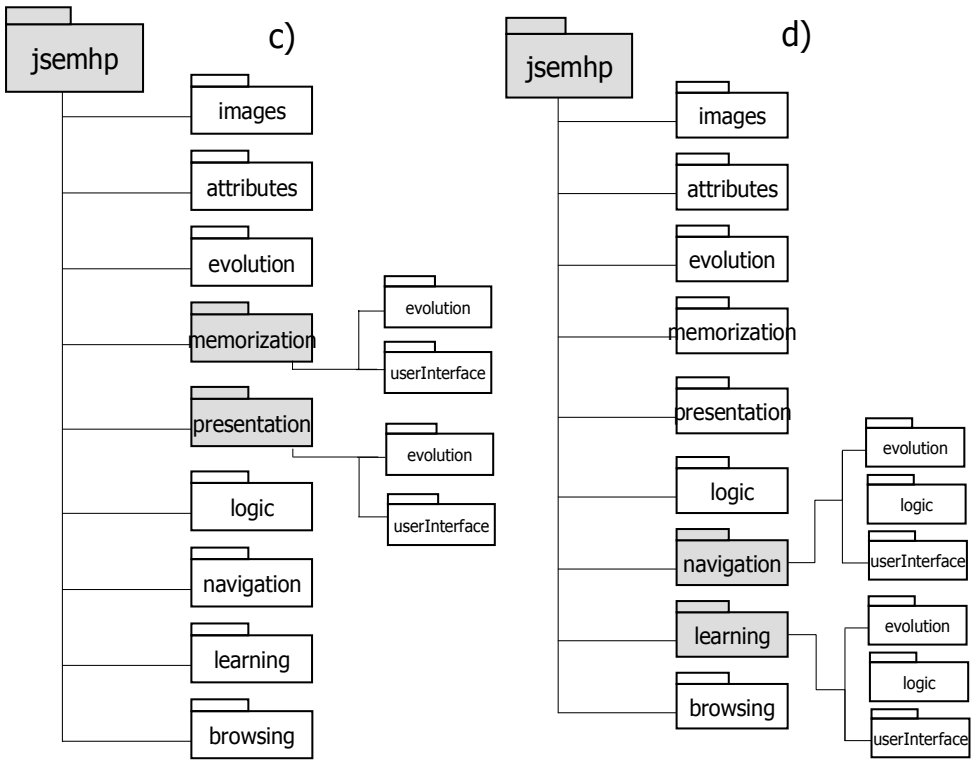


Figura 17: Estructura de paquetes

El paquete *images* (a) contiene los iconos gráficos usados en los menús y barras de herramientas de la interfaz del prototipo.

El paquete *evolution* (a) contiene las clases e interfaces implementadas para dotar de capacidad evolutiva a la herramienta. Algunos de los elementos más importantes incluidos en este paquete son: *MetaSystem*, *EvolutionarySystem*, *Restriction*, *EvAction*, etc.

El paquete *attributes* (b) contiene las clases e interfaces utilizadas para poder asignar atributos a los distintos elementos del modelo. Su principal elemento es la interfaz *AttributeValuePairs*. Dentro de este paquete existe un subpaquete denominado *userInterface* que tiene clases para gestionar los atributos en la interfaz de usuario.

El paquete *logic* (b) engloba las clases e interfaces usadas para construir expresiones lógicas complejas. Contiene, entre otros, los siguientes elementos: *ComplexLogicExpression*, *LogicExpression*, *LogicPredicate*, *LargerThan*, *LogicTerm*, etc. También incluye un subpaquete *userInterface* para el tratamiento de las expresiones lógicas a nivel de interfaz de usuario.

El paquete *browsing* (b) reúne las clases encargadas de la navegación del usuario. Por lo tanto incluye *CSbKnowledge*, *CSbTraditional*, *UserModel*, etc. De nuevo contiene un subpaquete *userInterface* con clases para gestionar la interfaz del usuario durante su navegación.

El paquete *memorization* (c) contiene todas las clases e interfaces necesarias para definir una estructura conceptual de memorización. Algunas de sus elementos más importantes son: *CSm*, *CSmDefault*, *CSmNode*, *FuncAssocEdge*, etc. Incluye dos subpaquetes, uno para gestionar la *CSm* a nivel de interfaz de usuario (*userInterface*) y otro para hacerla

evolucionar (*evolution*). Este último incluye acciones evolutivas y restricciones propias del sistema de memorización.

El paquete *presentation* (c) integra los elementos necesarios para crear presentaciones a partir de una estructura conceptual de memorización. Cabe destacar la clase *CSpDefault* y la interface *CSp*. También incluye los subpaquetes *userInterface* y *evolution*. En este caso, las acciones evolutivas y restricciones incluidas en el subpaquete *evolution* son específicas para una presentación.

El paquete *navigation* (d) engloba las clases e interfaces necesarias para definir y hacer evolucionar una estructura conceptual de navegación. Entre sus elementos más significativos podemos destacar *CSn*, *CSnDefault* y *OrderRule*. Además de los subpaquetes de interfaz y evolución, existe un subpaquete *logic* que incluye clases que representan predicados lógicos específicos de las reglas de orden como son *Before* y *Previous*.

El paquete *learning* (d) reúne las clases e interfaces implementadas para definir y hacer evolucionar un conjunto de reglas de aprendizaje. Algunos de sus elementos más importantes son: *UpdateRule*, *KnowledgeRule*, *CSl*, *CSlDefault*, etc. Contiene, al igual que *navigation*, tres subpaquetes: *userInterface*, *evolution* y *logic*. En el subpaquete *evolution* encontramos acciones evolutivas para modificar las reglas de conocimiento y actualización (*AddKnowledgeRule*, *RemoveKnowledgeRule*,...). En el subpaquete *logic* encontramos predicados lógicos propios de las reglas de actualización (*UpdatePred*, *FixAbs*, *IncRel*,...).

4.4 Diagramas de secuencia y colaboración

El diagrama de secuencia del sistema varía según se esté trabajando en la creación/evolución del sistema (figura 18) o navegándolo (figura 19). A parte de que el actor es distinto, el autor en la primera secuencia y el usuario en la segunda, la principal diferencia radica en la intervención o no del meta-sistema (éste sólo actúa en el primer caso).

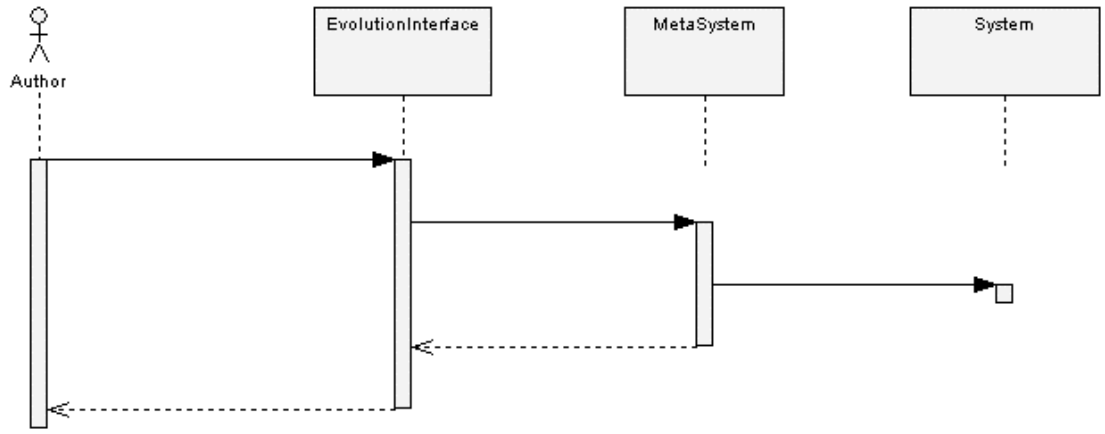


Figura 18: Secuencia del sistema (autor)

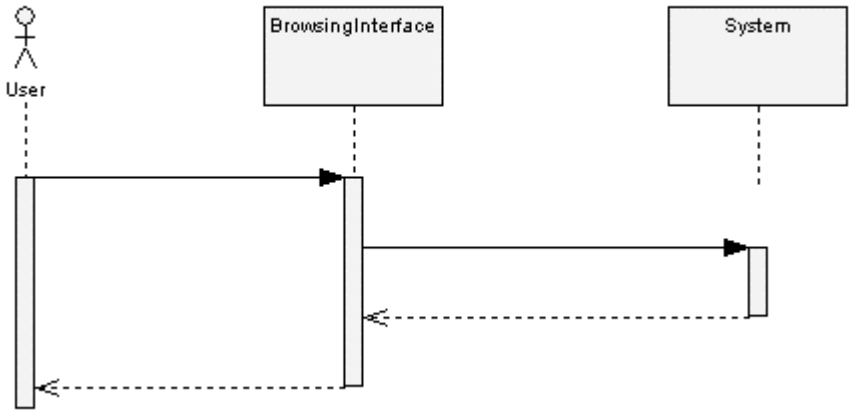


Figura 19: Secuencia del sistema (usuario)

A continuación se incluyen dos diagramas de colaboración que ilustran las tareas principales del sistema. El primer diagrama (figura 20) muestra, de forma general, lo que ocurre cuando el autor realiza un evento que se traduce en la ejecución de una acción evolutiva. El segundo diagrama (figura 21) revela el paso de mensajes que tiene lugar cuando el usuario selecciona un ítem en la interfaz de navegación en modo por conocimiento.

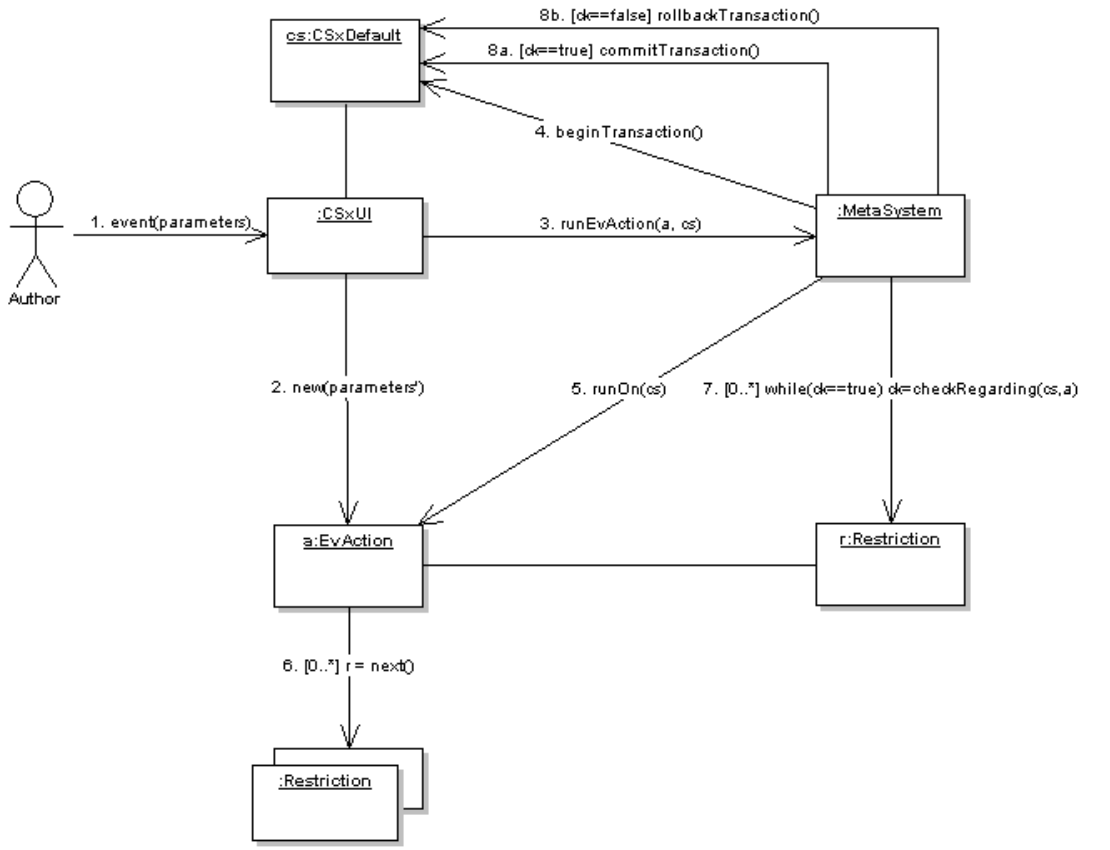


Figura 20: Ejecución de una acción evolutiva

El autor solicita una modificación realizando un determinado evento (1) en la interfaz de evolución de la estructura conceptual actual (CSxUI puede ser particularizado como *CSmUI*, *CSpUI*, *CSnUI* o *CSlUI*).

En consecuencia se crea una acción evolutiva del tipo necesario para llevar a cabo la modificación (2), y se solicita su ejecución a *MetaSystem* (3). El meta-sistema envía la orden de comenzar una transacción (4) a la estructura conceptual que se va a modificar (CSxDefault puede instanciarse como *CSmDefault*, *CSpDefault*, *CSnDefault* o *CSlDefault*) y pide a la propia acción evolutiva que se ejecute (5).

La acción evolutiva consulta sus restricciones (6), y el cumplimiento de cada una de éstas se comprueba sobre la estructura modificada (7). Si todas las restricciones se satisfacen, el meta-sistema ordena a CSxDefault que termine la transacción (8a), en caso contrario le hace volver a su estado anterior (8b).

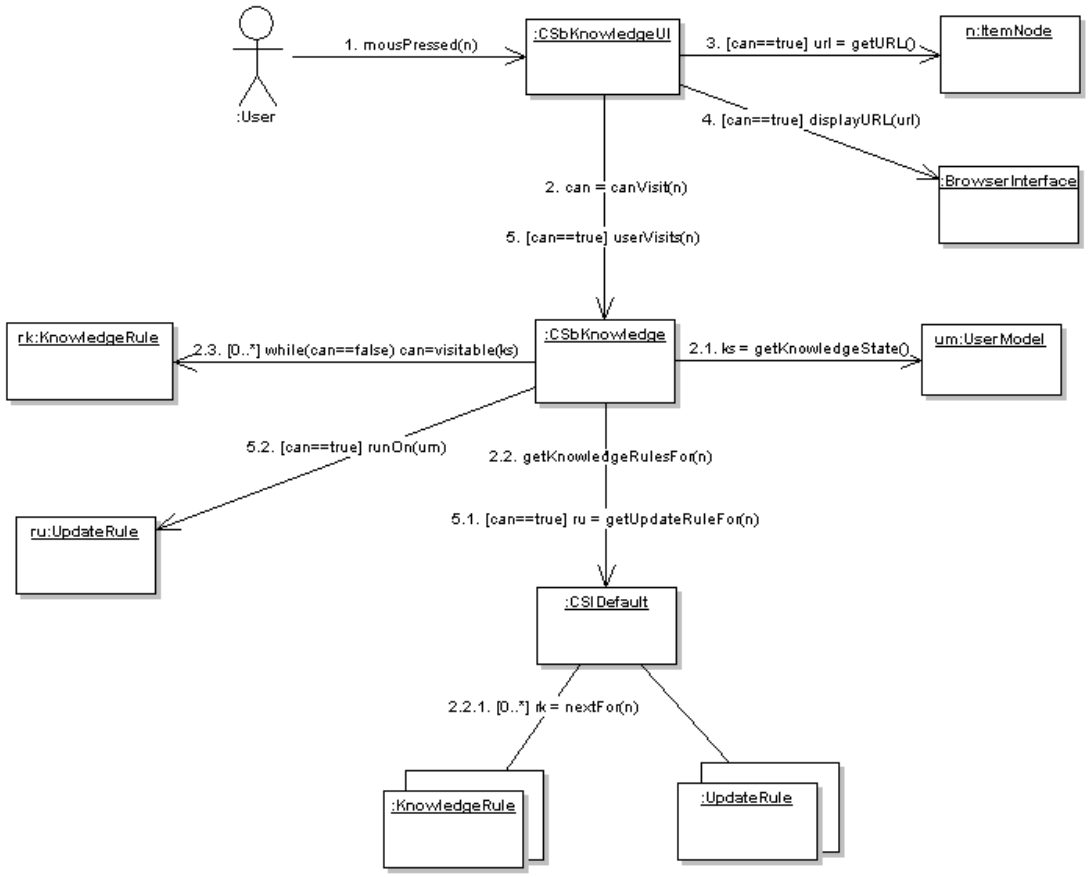


Figura 21: Selección de un ítem en navegación por conocimiento

Cuando el usuario presiona el botón izquierdo del ratón sobre un ítem (1) en la interfaz de navegación por conocimiento (*CSbKnowledgeUI*), ésta pregunta a *CSbKnowledge* si dicho ítem es accesible para el usuario (2). Para ello se consulta el estado de conocimiento del usuario en su modelo (2.1), se obtienen las reglas de conocimiento del ítem (2.2, 2.2.1) en *CSlDefault* y se evalúan (2.3) para comprobar si en al menos una se satisface el antecedente de accesibilidad. Si la visita se permite, se averigua la *url* del ítem (3) y se le pide al navegador que la visualice (4). Después se busca (5.1) y ejecuta (5.2) la regla de actualización del ítem visitado.

5. UN EJEMPLO DE USO

En esta sección se muestra el prototipo JSEM-HP en funcionamiento. Las ventanas que se presentan han sido capturadas durante la construcción (sección 5.1) y navegación (sección 5.2) de un sistema hipermedia adaptativo cuyo dominio de conocimiento es el paradigma orientado a objetos. Pueden verse más ejemplos de uso del prototipo en [Molina, 03] [Medina, 02d].

5.1 Creación del sistema (autor)

La construcción del sistema hipermedia adaptativo es una tarea del autor que se divide en cuatro fases: memorización, presentación, navegación y aprendizaje. En el ejemplo se muestran los pasos que el autor realiza durante cada una de estas fases[1], visualizando en cada paso, la acción realizada por el autor y las consecuencias de la misma.

5.1.1 Fase de memorización

Paso I: Para empezar a trabajar abrir el editor de estructuras conceptuales: CS Editor.

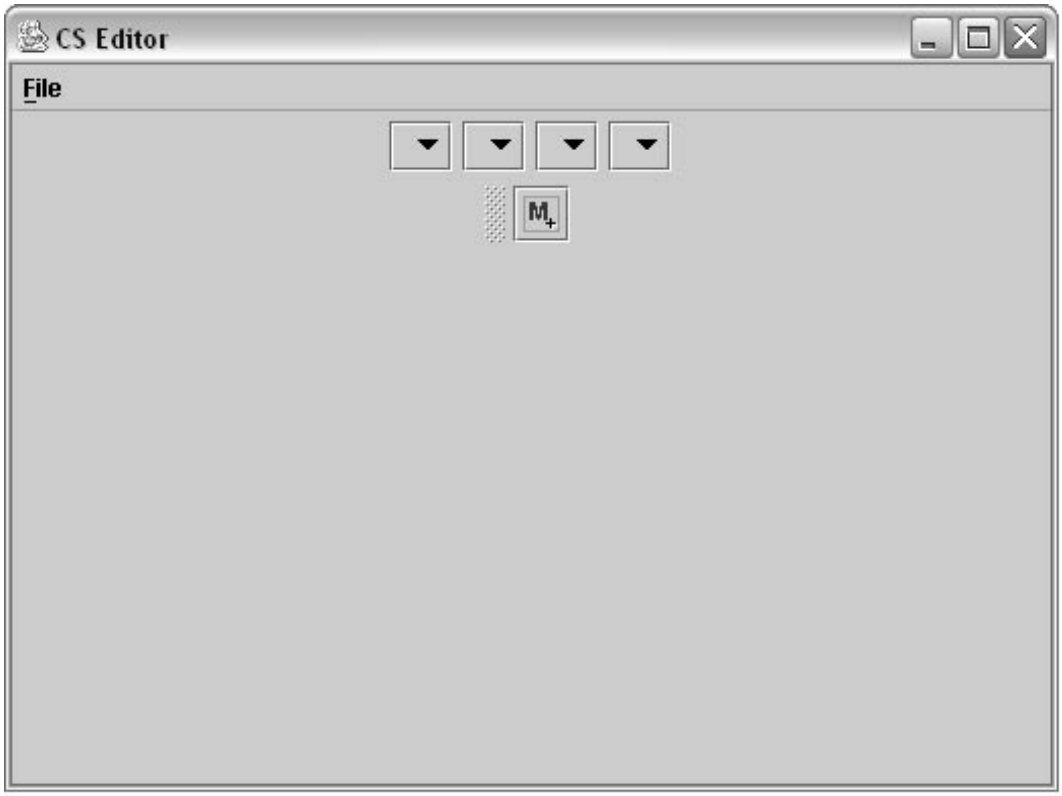


Figura 22: Interfaz para editar estructuras conceptuales

[1] Lo que aquí se presenta es un ejemplo. Los pasos no tienen porque realizarse exactamente en el orden que aquí se hace. Además, las fases no son secuenciales, sino iterativas, de forma que, en cualquier momento, se puede volver a una fase anterior.

Paso II: Pulsar el botón *M+* para crear una nueva estructura conceptual de memorización.

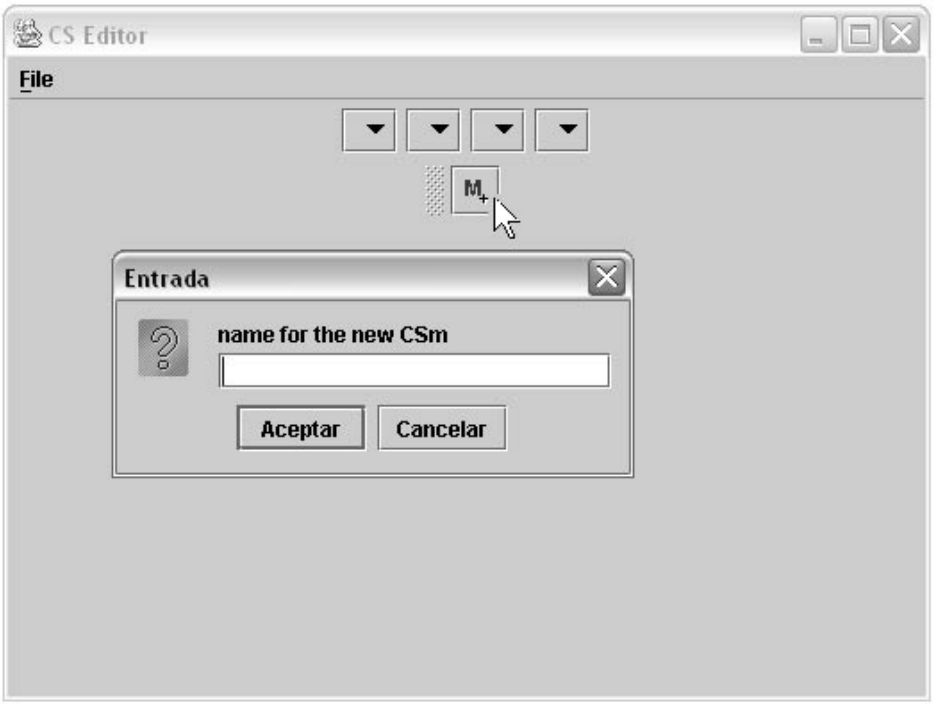


Figura 23: Crear una nueva EC_M

Paso III: Escribir el nombre de la nueva estructura conceptual de memorización, *object-orientation* en el ejemplo.

Figura 24: Nombrar la EC_M

Paso IV: Empezar a trabajar con la estructura conceptual de memorización.

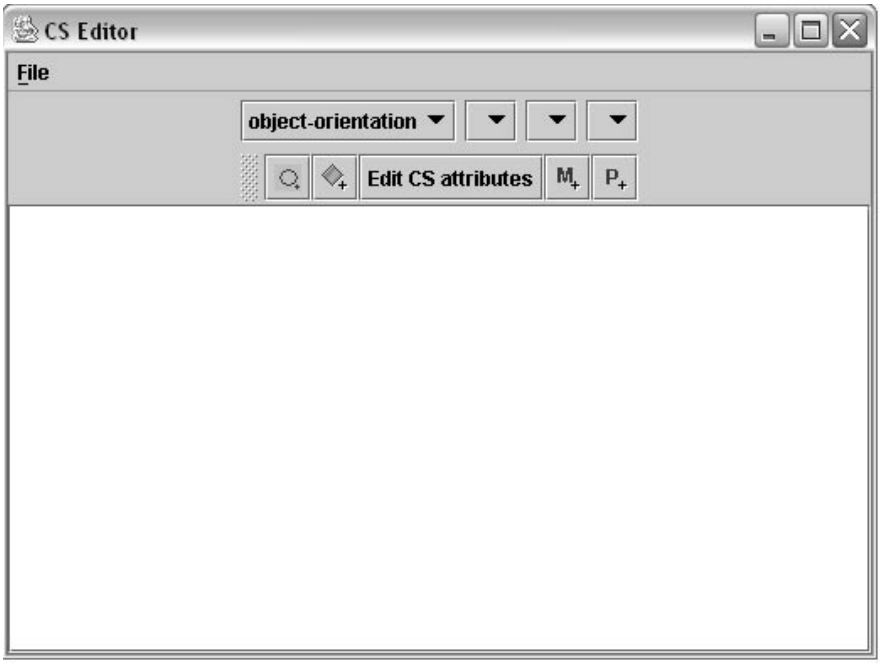


Figura 25: Interfaz para definir la EC_M

Paso V: Para crear un nuevo concepto pulsar el botón derecho del ratón sobre la posición donde se desea insertarlo, y seleccionar la opción *Insert Concept* en el menú que aparece.

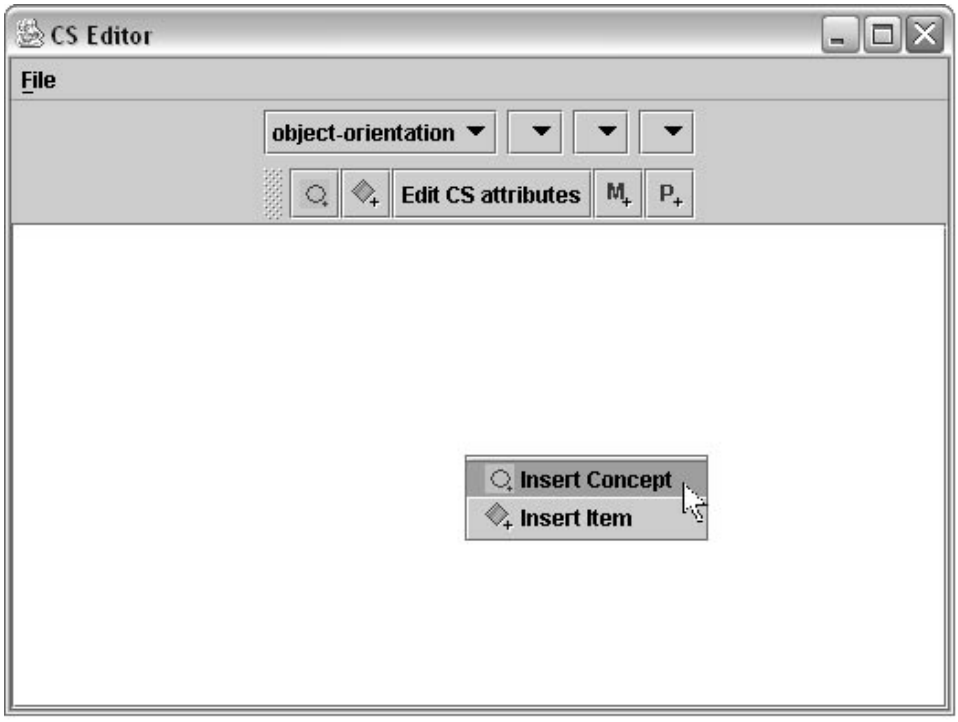


Figura 26: Insertar un concepto en una posición específica

También se puede usar directamente el botón  del menú superior. En tal caso, el concepto se inserta en la esquina superior izquierda. Luego se puede colocar en la posición que se desee seleccionando y arrastrando con el ratón.

Paso VI: Escribir el nombre del concepto en el cuadro que aparece para ello. Si se desea, también se puede escribir una descripción del concepto.

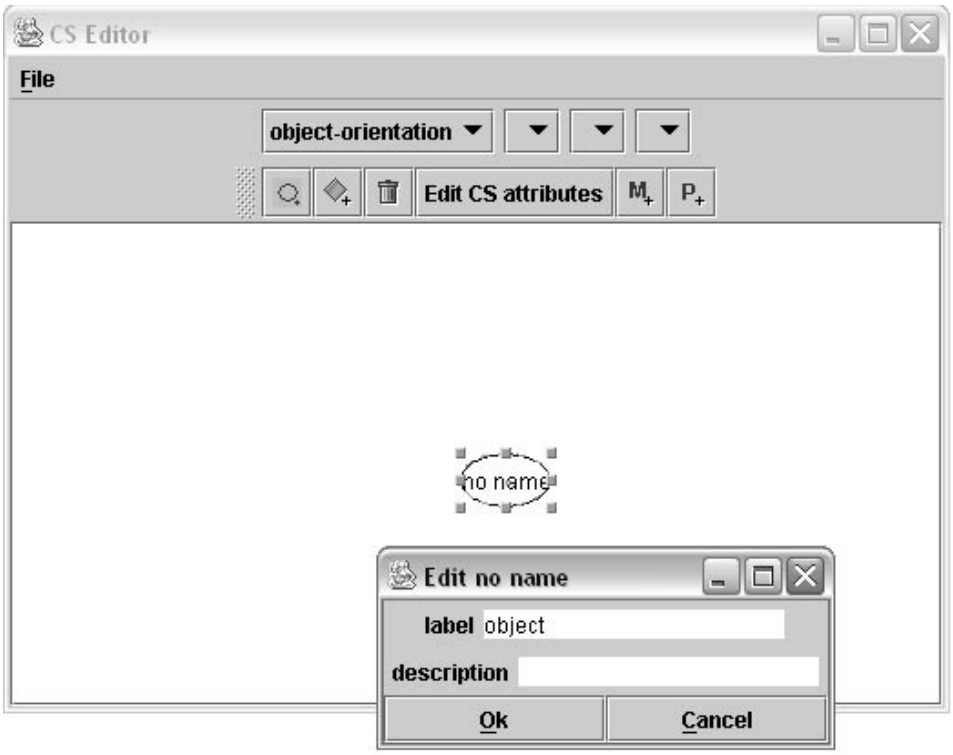


Figura 27: Nombrar el concepto insertado

Paso VII: Seguir creando conceptos. Si se intenta introducir otro concepto con el mismo nombre que uno ya existente el sistema genera un mensaje de error y rechaza la acción.

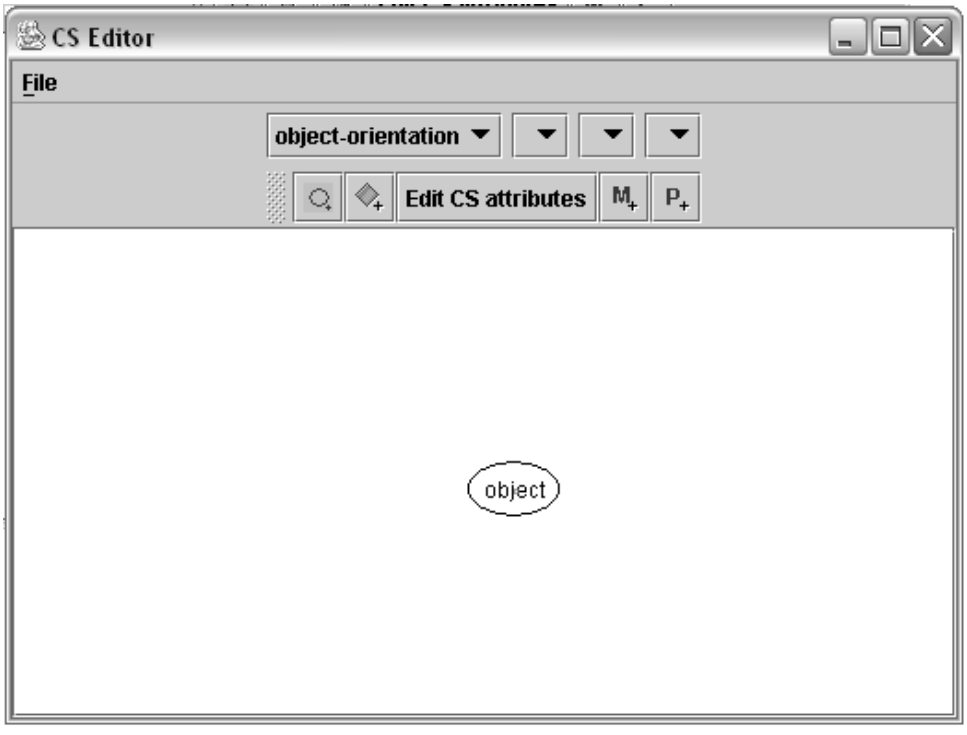


Figura 28: Concepto *object* insertado

Paso VIII: Establecer una relación conceptual entre dos conceptos, situando el cursor en el centro del concepto origen y arrastrando hasta el centro del concepto destino.

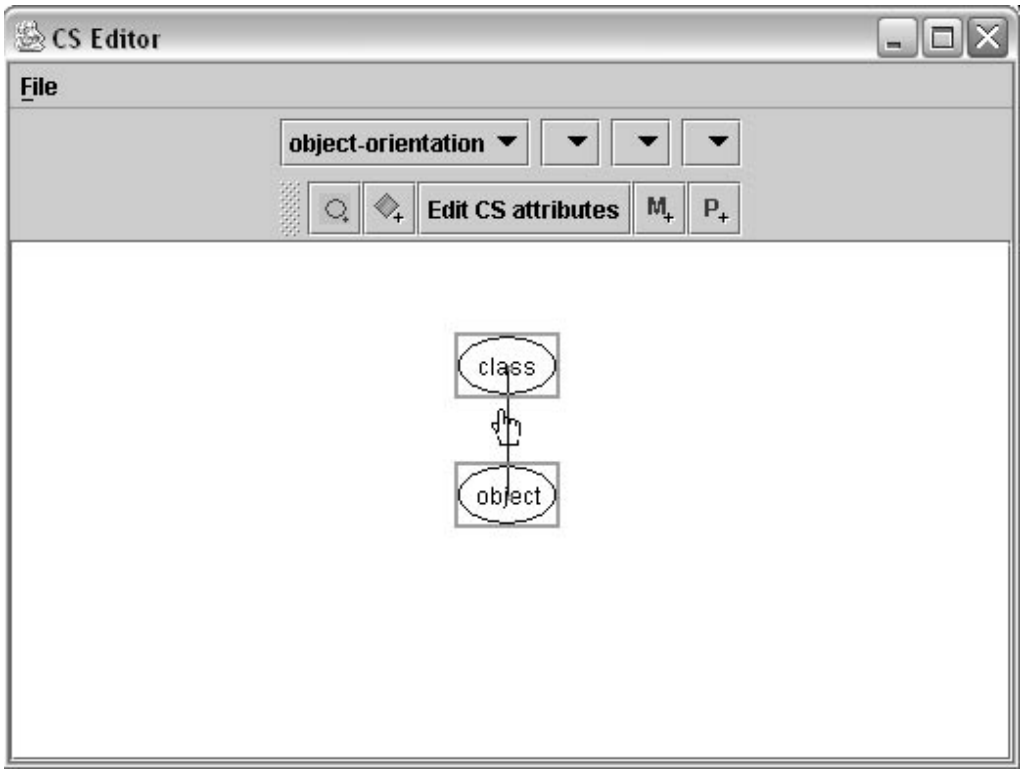


Figura 29: Relación conceptual entre *class* y *object*

Paso IX: Pulsar el botón derecho del ratón sobre la relación conceptual para editar (*Edit*) sus propiedades.

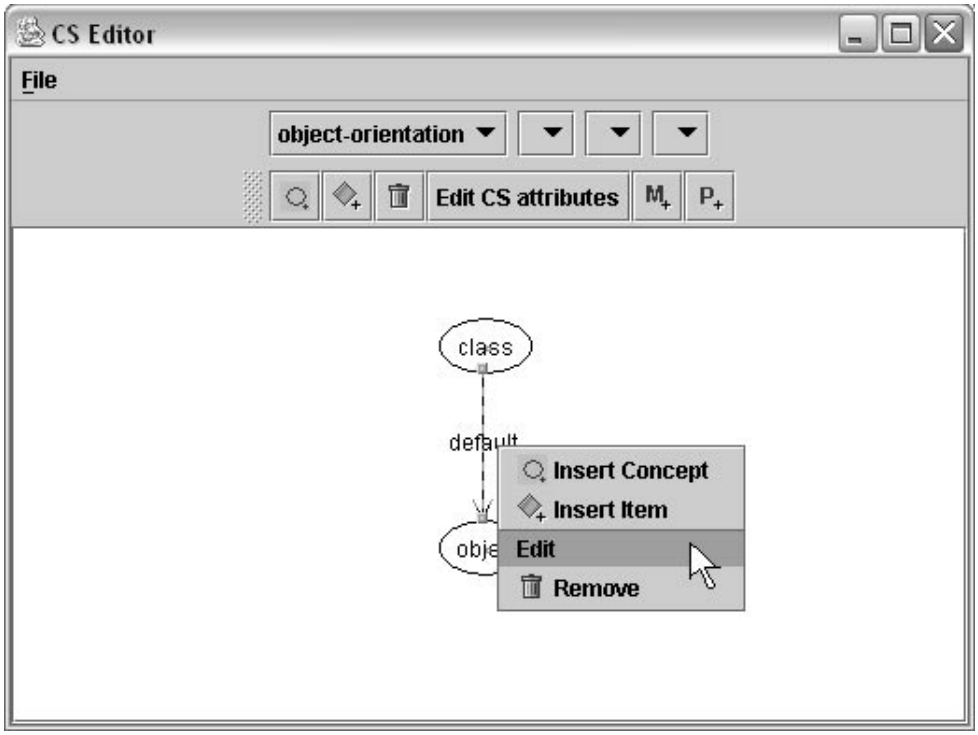


Figura 30: Editar las propiedades de la relación conceptual

Paso X: Elegir el tipo de relación conceptual en la lista desplegable que aparece.

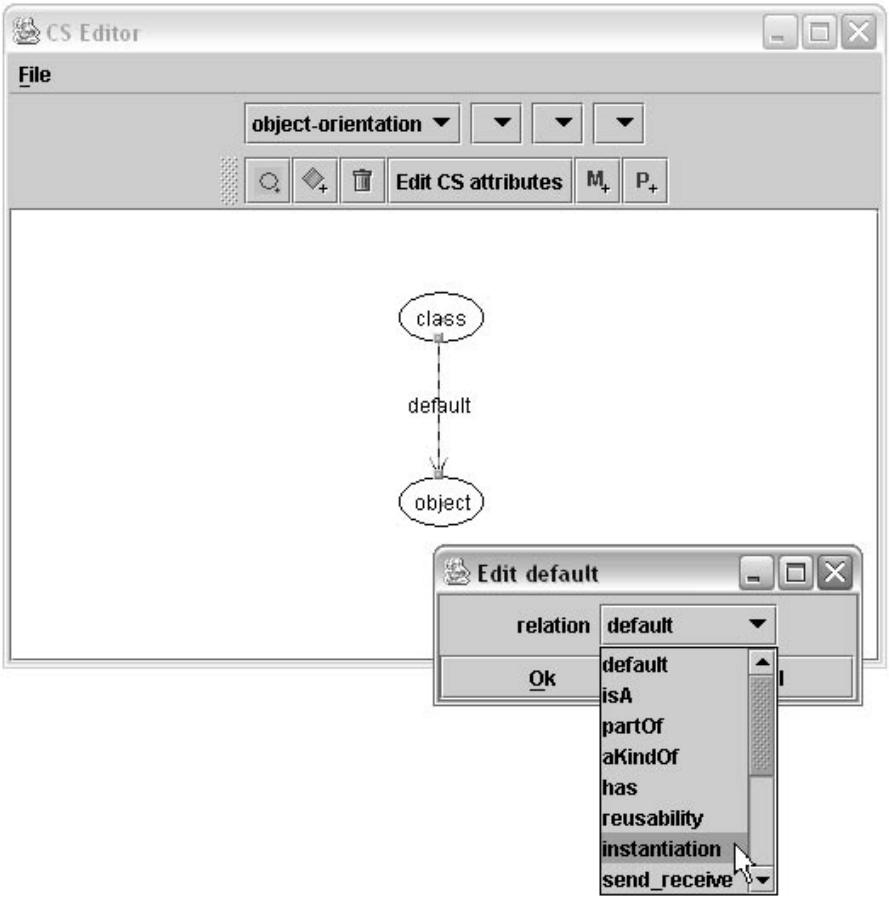


Figura 31: Elegir el tipo de la relación conceptual

Paso XI: Crear tantos conceptos y relaciones conceptuales como se desee.

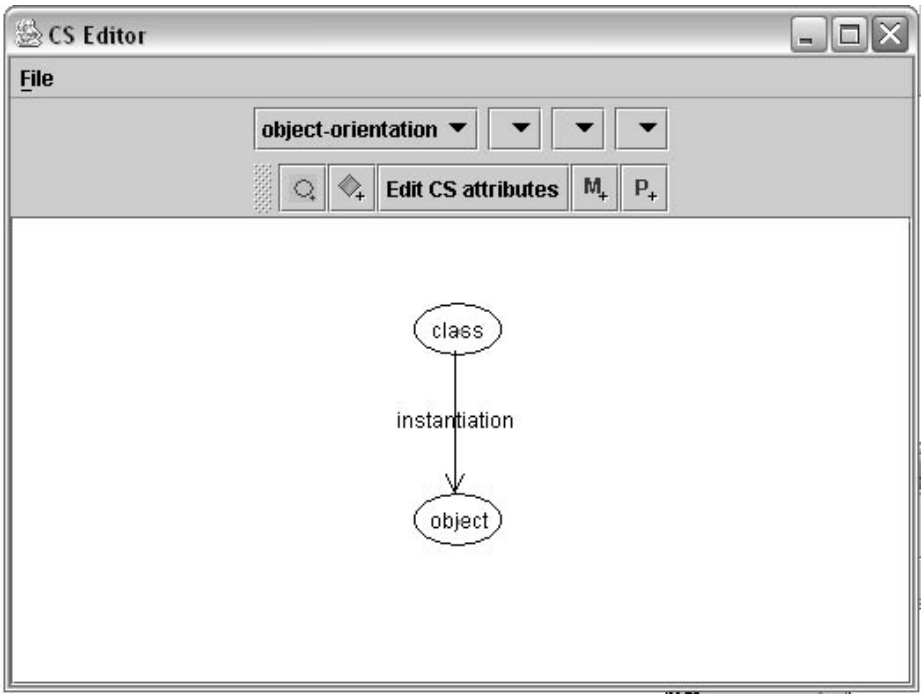


Figura 32: *Relación conceptual instantiation: class → object*

Paso XII: Pulsar *File → Save to file...* para guardar el trabajo realizado hasta el momento.

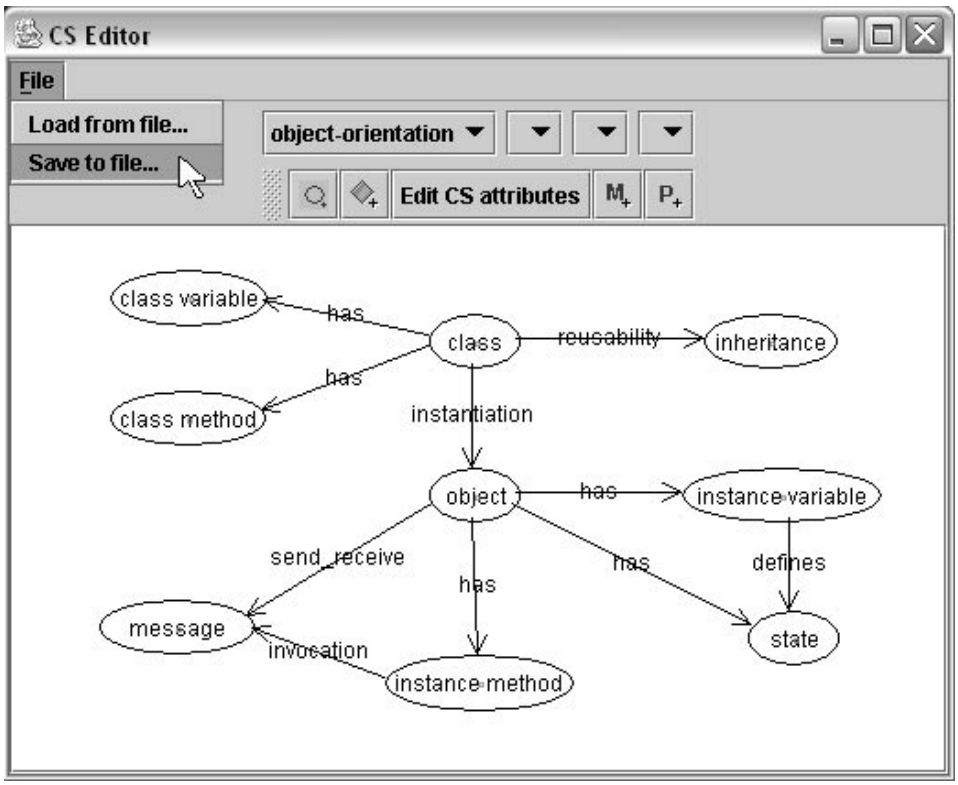


Figura 33: Salvando el trabajo

Paso XIII: Insertar un ítem usando el botón en el menú superior o sobre la posición de inserción deseada. Para editar las propiedades del ítem pulsar el botón derecho sobre él y elegir *Edit* en el menú que aparece.

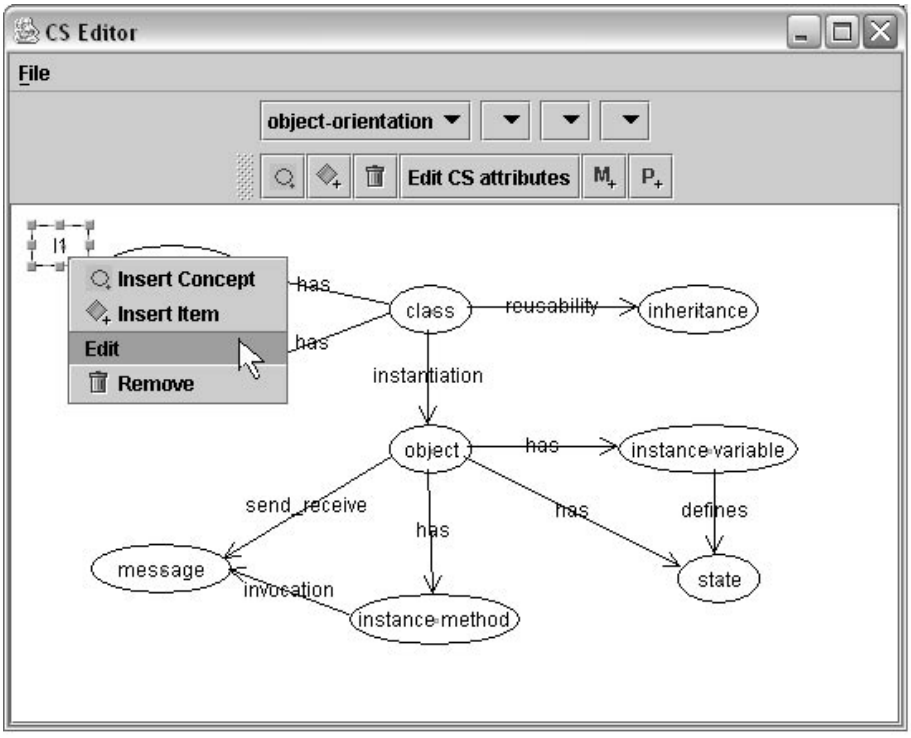


Figura 34: Crear un ítem de información

Paso XIV: Rellenar las propiedades del ítem en el cuadro de edición, indicando identificador, etiqueta, *url*, idioma, nivel de dificultad, medio, autor y fecha. Para el idioma, la dificultad y el medio, se elige un valor de una lista.

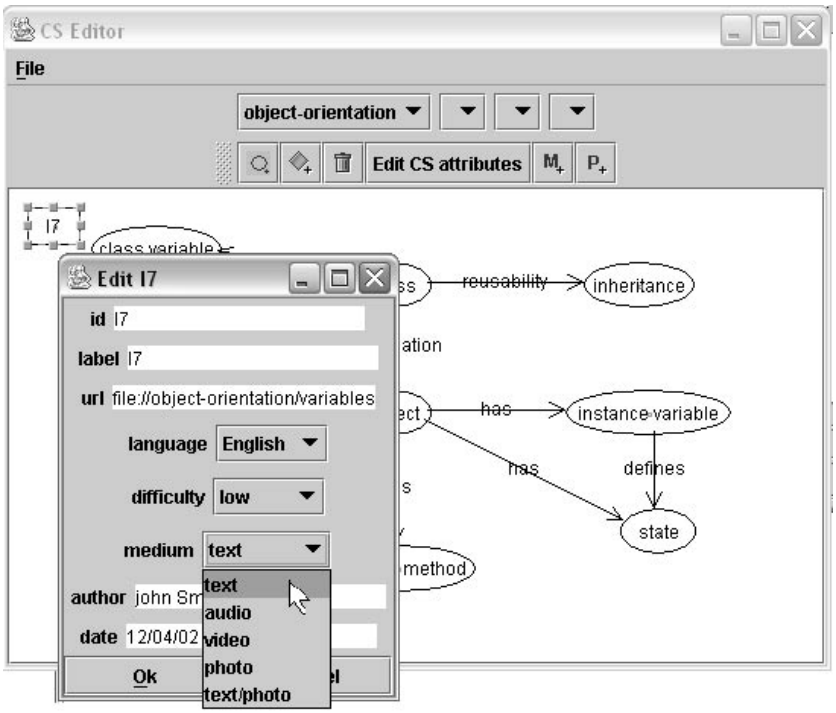


Figura 35: Establecer las propiedades del ítem

El sistema asigna por defecto el identificador del ítem siguiendo una numeración secuencial. Sin embargo, es posible modificar el identificador de un ítem, siempre que el valor dado no coincida con uno ya usado en la estructura actual.

Paso XV: Crear una asociación funcional entre un ítem y un concepto, pinchando en el centro del primero y arrastrando hasta el centro del segundo.

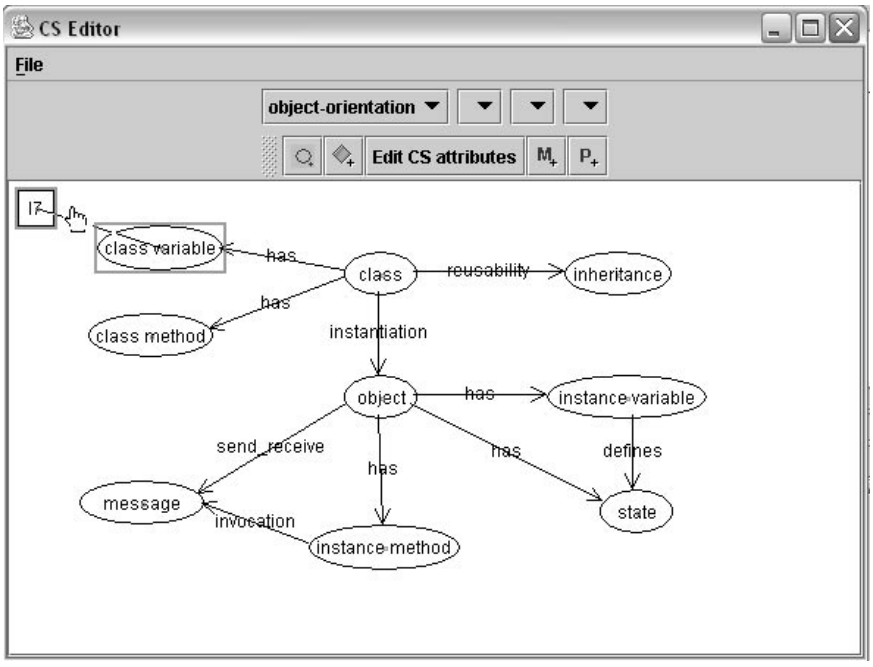


Figura 36: Crear una asociación funcional entre el ítem I7 y el concepto *class variable*

Paso XVI: Establecer el rol de la asociación funcional pulsando el botón derecho sobre la relación y eligiendo un valor de la lista que aparece cuando editamos las propiedades.

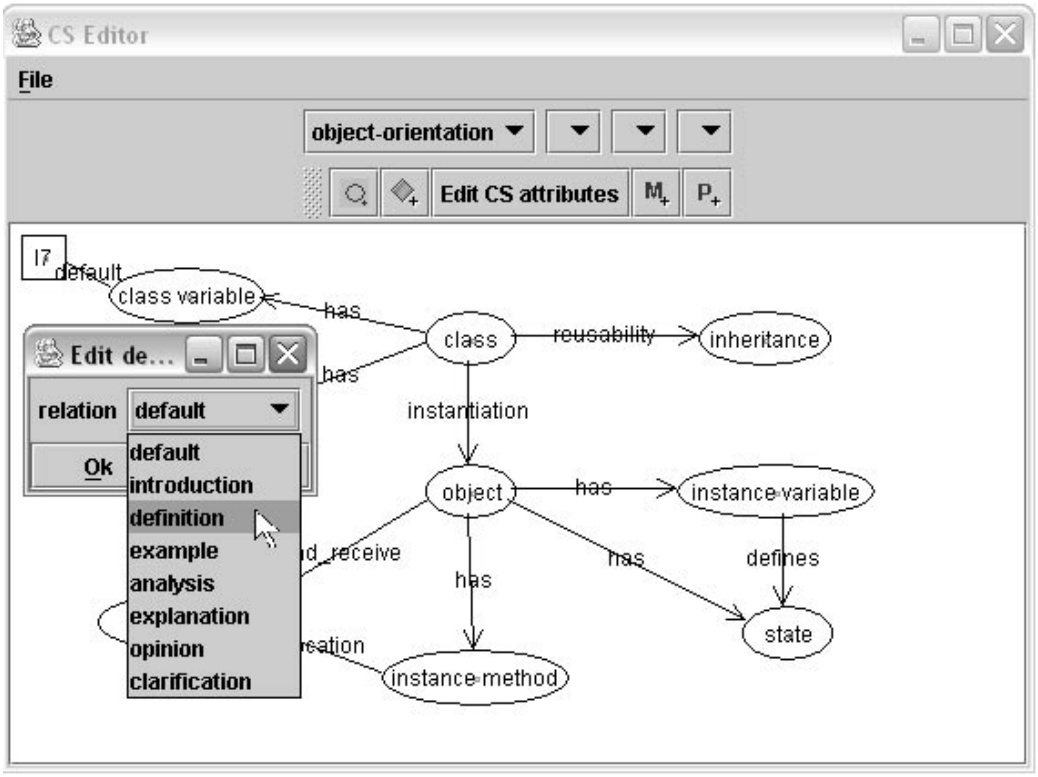


Figura 37: Asignar el rol *definition* a la asociación funcional entre I7 y *class variable*

Paso XVII: Seguir insertando conceptos, ítems, relaciones conceptuales y asociaciones funcionales hasta completar la estructura conceptual de memorización final.

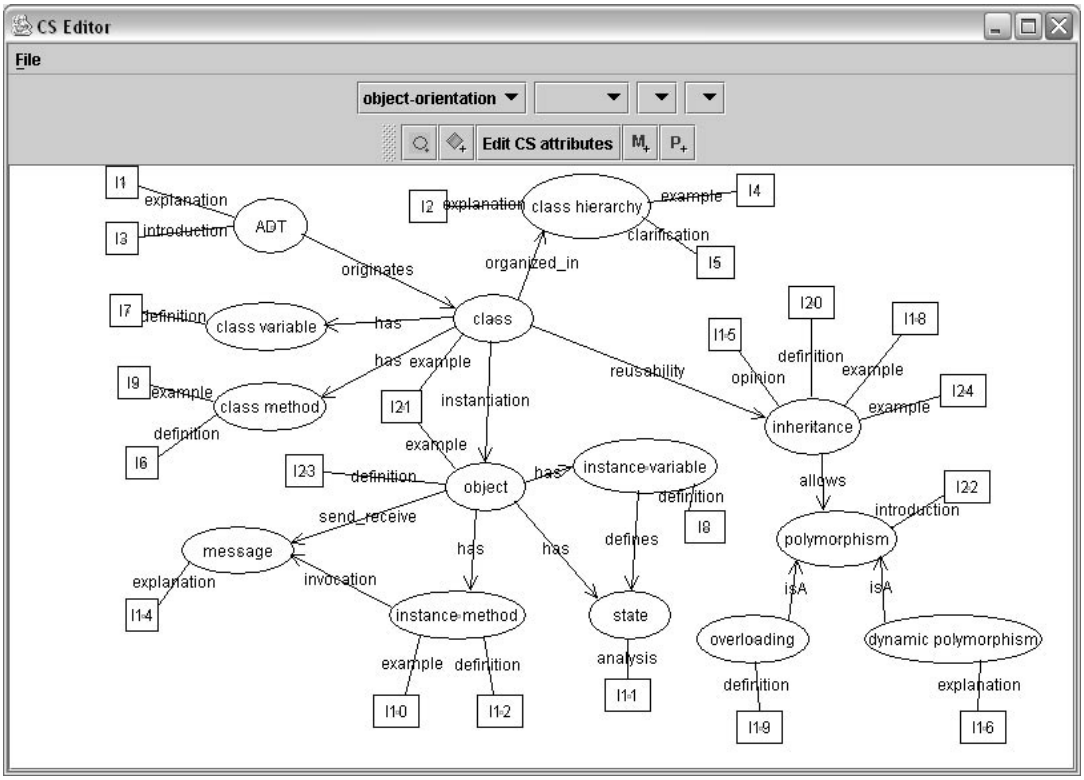


Figura 38: La estructura conceptual de memorización

5.1.2 Fase de presentación

Paso I: Teniendo una estructura conceptual de memorización abierta, pulsar el botón *P+* para crear una presentación a partir de ella. Escribir el nombre de la nueva presentación en el cuadro que aparece.

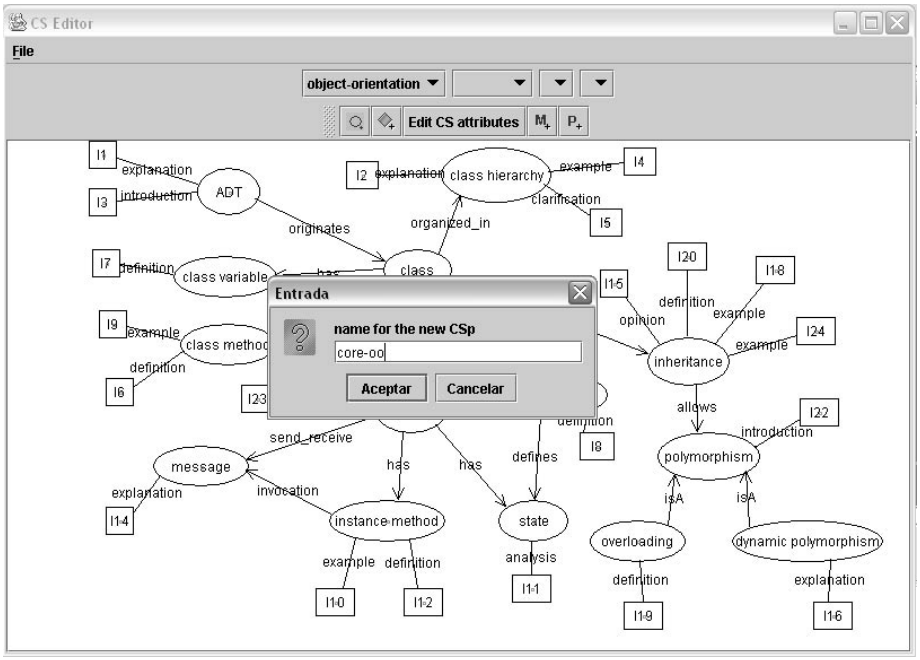


Figura 39: Creando una presentación llamada *core-oo*

Paso II: Ocultar los conceptos que se desee pulsando el botón derecho sobre ellos y seleccionando la opción *Hide* en el menú que aparece. Para recuperarlos pulsar *Show*. También se puede seleccionar el concepto y usar los botones del menú superior  .

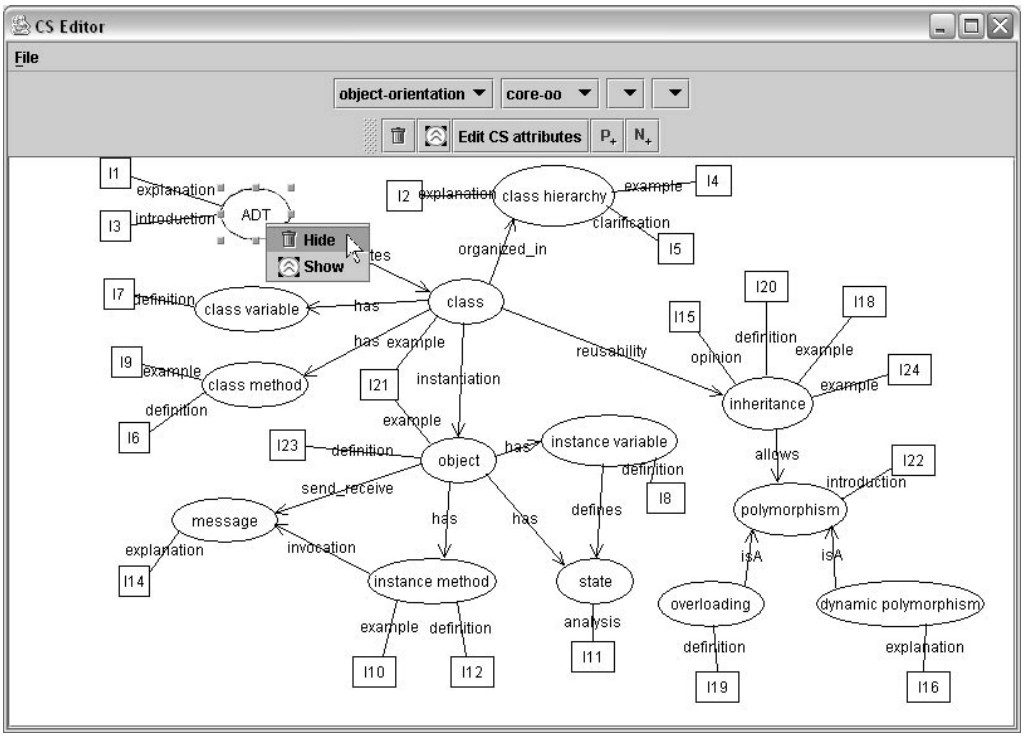


Figura 40: Ocultar el concepto *ADT*

Paso III: Seguir ocultando conceptos, ítems y relaciones hasta obtener la presentación deseada.

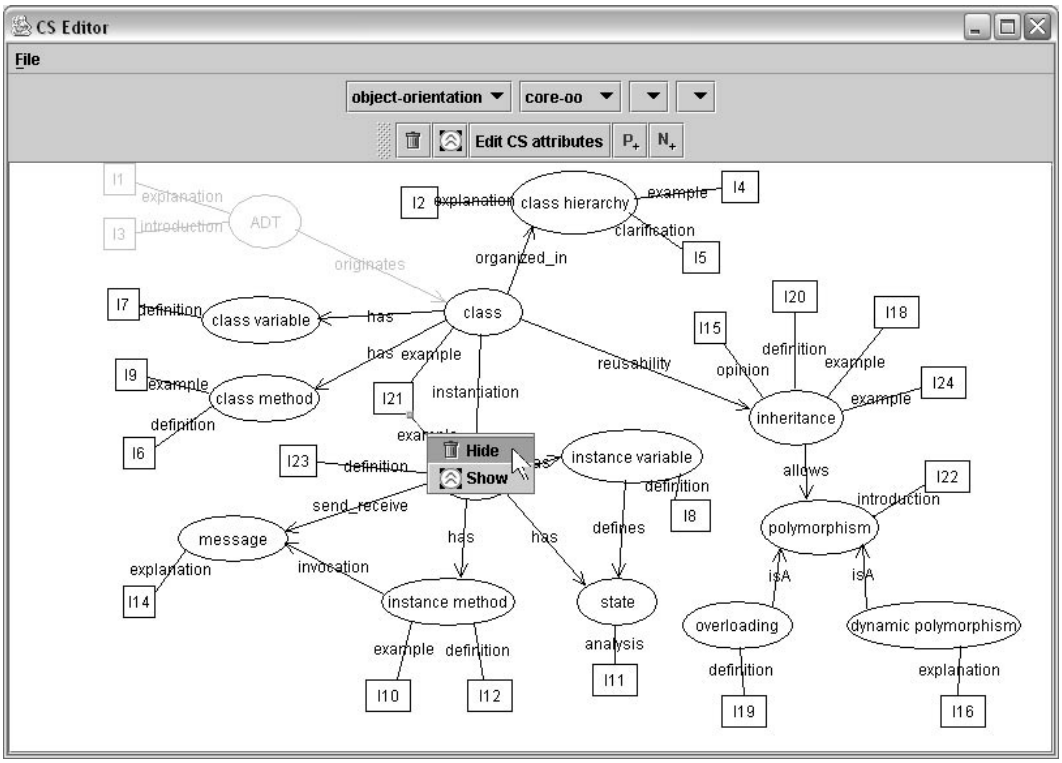


Figura 41: Ocultar la asociación funcional del ítem I21 con el concepto *object*

5.1.3 Fase de navegación

Paso I: Teniendo una presentación abierta pulsar el botón *N+* para definir una estructura conceptual de navegación a partir de ella. Escribir el nombre de la nueva EC_N en el cuadro que aparece, *core-oo-n1* en el ejemplo.

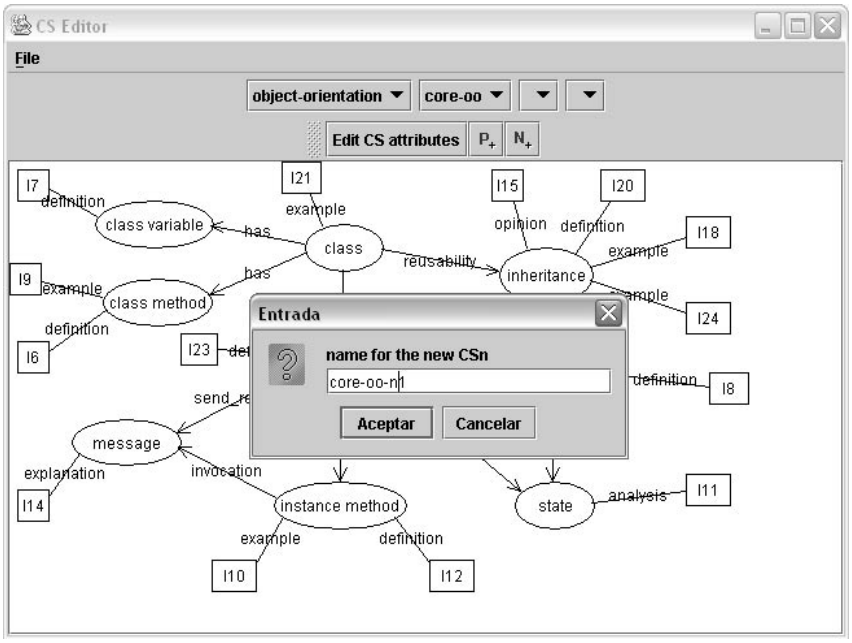


Figura 42: Crear una nueva estructura conceptual de navegación

Paso II: Ampliar la navegabilidad de las relaciones conceptuales que se desee, pulsando el botón derecho sobre la relación y seleccionando *Extend navigability*. Después se puede reducir con la opción *Reduce navigability*. Las relaciones con navegabilidad extendida se mostrarán con doble flecha.

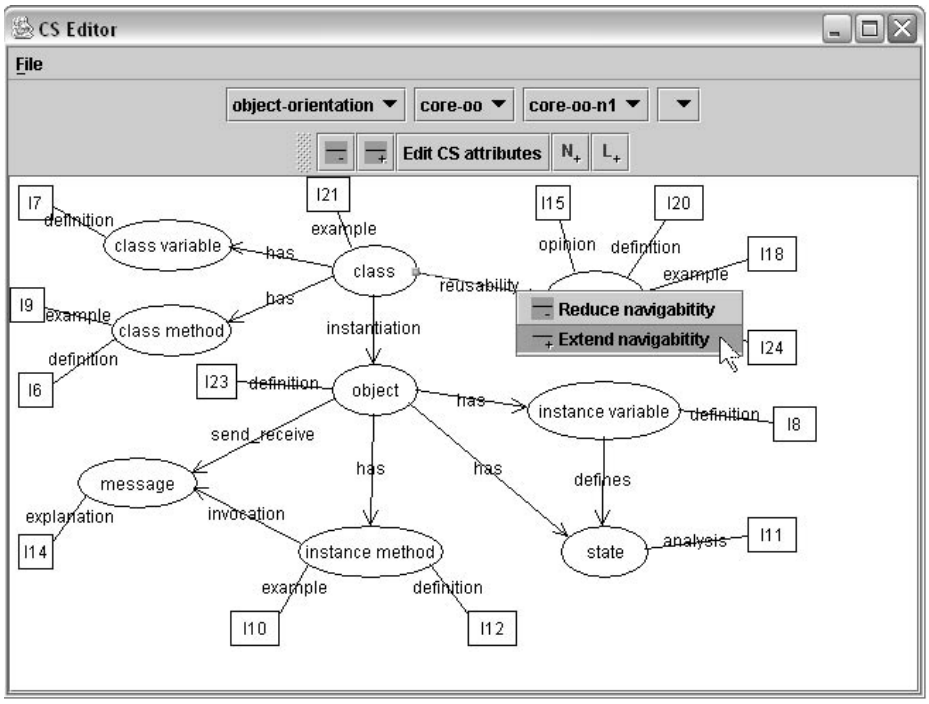


Figura 43: Extender la navegabilididad de la relación conceptual *reusability*

También se pueden usar los botones  que aparecen en el menú superior.

Paso III: Obtener las reglas de orden creadas por defecto para un ítem, pulsando botón derecho sobre éste y seleccionando la opción de edición (*Edit order rules*).

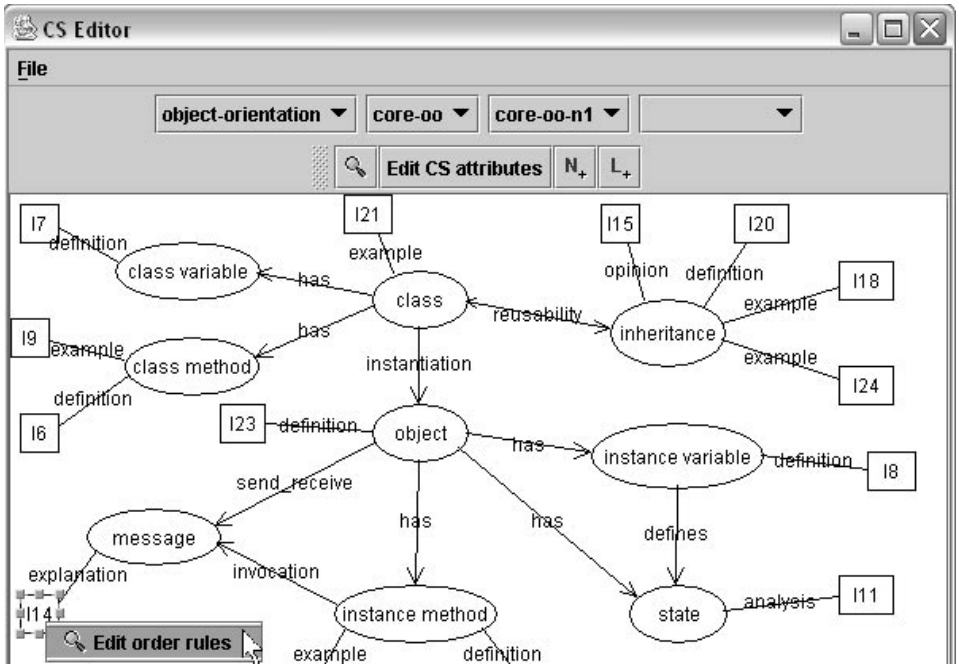


Figura 44: Editar las reglas de orden para el ítem I14

Paso IV: Observe que existe una regla de orden por cada relación conceptual que llega al concepto *message* al que I14 se asocia funcionalmente. Cada regla exige la visita previa a alguno de los ítems asociados al concepto origen de la relación.

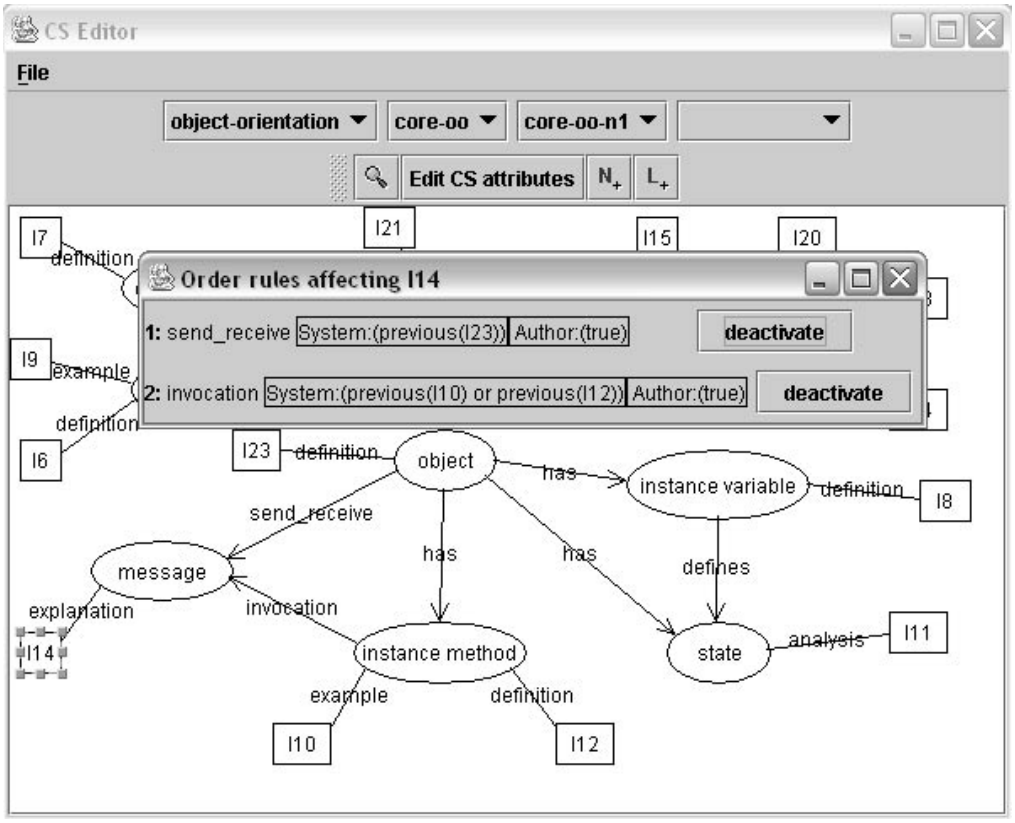


Figura 45: Editar las reglas de orden para el ítem I14

Paso V: Reemplazar el predicado *true* de la fórmula de autor de la primera regla por el predicado *before(I9)*. De esta forma se exige haber visitado I9 antes de visitar I14 a través de la relación conceptual *send_receive*.

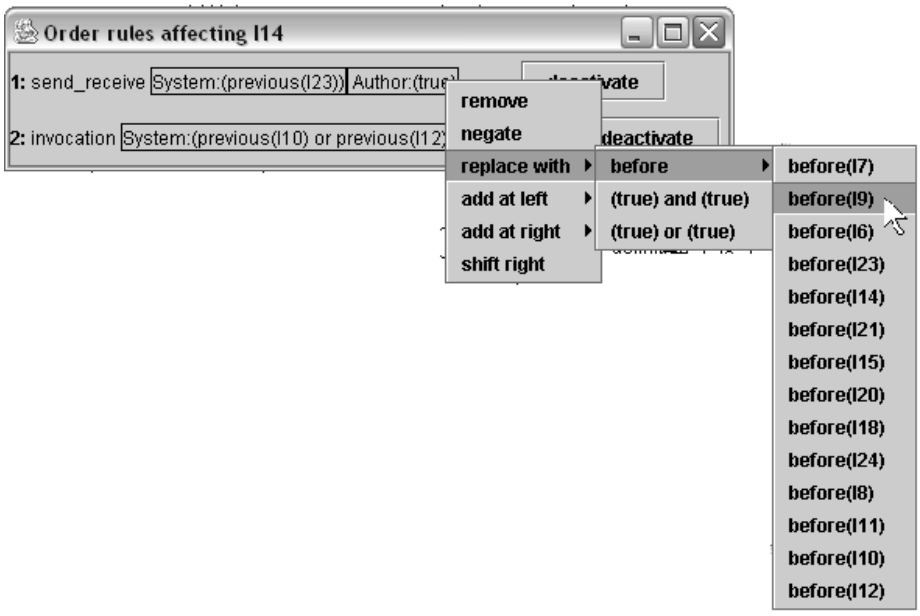


Figura 46: Modificar la formula de autor en Ro^1(I14)

Para modificar de esta forma el predicado, es necesario situarse encima y pulsar el botón derecho, seleccionando después las opciones *replace with* → *before* → *before(I9)*.

Paso VI: Reemplazar el predicado *true* de la fórmula de autor de la segunda regla por el predicado compuesto *before(I8) and before(I11)*.

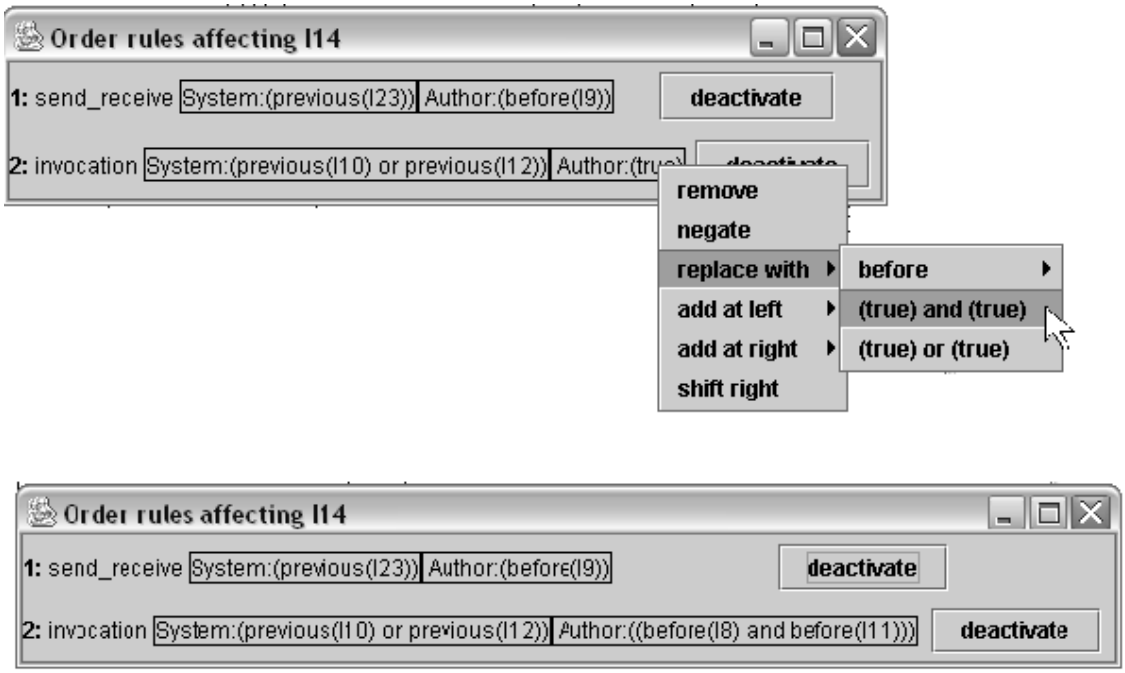


Figura 47: Modificar la formula de autor en Ro^2(I14)

En la fórmula de autor se pueden escribir predicados compuestos tan complejos como se desee haciendo uso de las opciones *replace with* → *(true) and (true)* y *replace with*→ *(true) or (true)*. Después cada predicado *true* se reemplaza por un predicado simple o nuevamente compuesto.

También se puede negar un predicado con la opción *negate*, añadir un nuevo predicado a la izquierda (*add at left*) o a la derecha (*add at right*) del predicado actual, o reordenar los predicados usando *shift right*.

Observe en la figura 46, que la lista de predicados posibles para la fórmula de autor sólo incluye predicados de tipo *before* impuestos sobre ítems de la estructura actual. La fórmula del sistema exige visitas inmediatas (*previous*) y es fija, es decir no se puede modificar.

Sin embargo, en determinadas circunstancias es posible desactivar una regla de orden pulsando el botón *deactivate* situado junto a ella. Después para activar la regla basta con pulsar el botón *activate* que aparece junto a ella cuando se encuentra en estado inactivo.

5.1.4 Fase de aprendizaje

Paso I: Teniendo una estructura conceptual de navegación abierta pulsar el botón *L+* para definir una estructura conceptual de aprendizaje a partir de ella. Escribir el nombre de la nueva EC_L en el cuadro que aparece con tal fin, *core-oo-n1-l1* en el ejemplo.

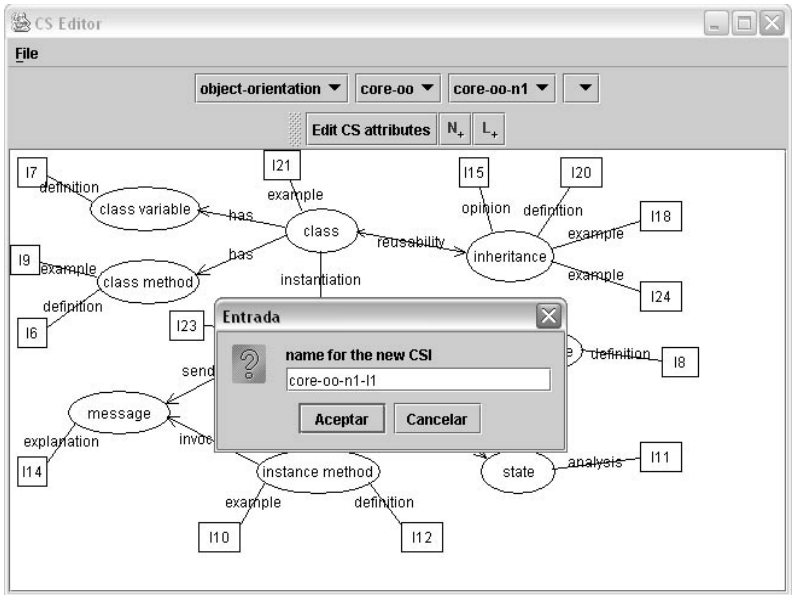


Figura 48: Crear una nueva EC_L

Observe que en el menú superior aparecen cuatro listas desplegables, que corresponden a cada una de las cuatro fases de diseño, y que permiten elegir:

1. la estructura conceptual de memorización (*object-orientation*),
2. una presentación de la EC_M seleccionada en la lista 1 (*core-oo*),
3. una navegación creada a partir de la EC_P seleccionada en la lista 2 (*core-oo-n1*),
4. y, un aprendizaje definido sobre la EC_N elegida en la lista 3 (vacío en este momento).

Paso II: Pulsar el botón *Edit CS attributes* para etiquetar la estructura de aprendizaje: rangos de experiencia (de navegación y en la materia) y subdominio de conocimiento.

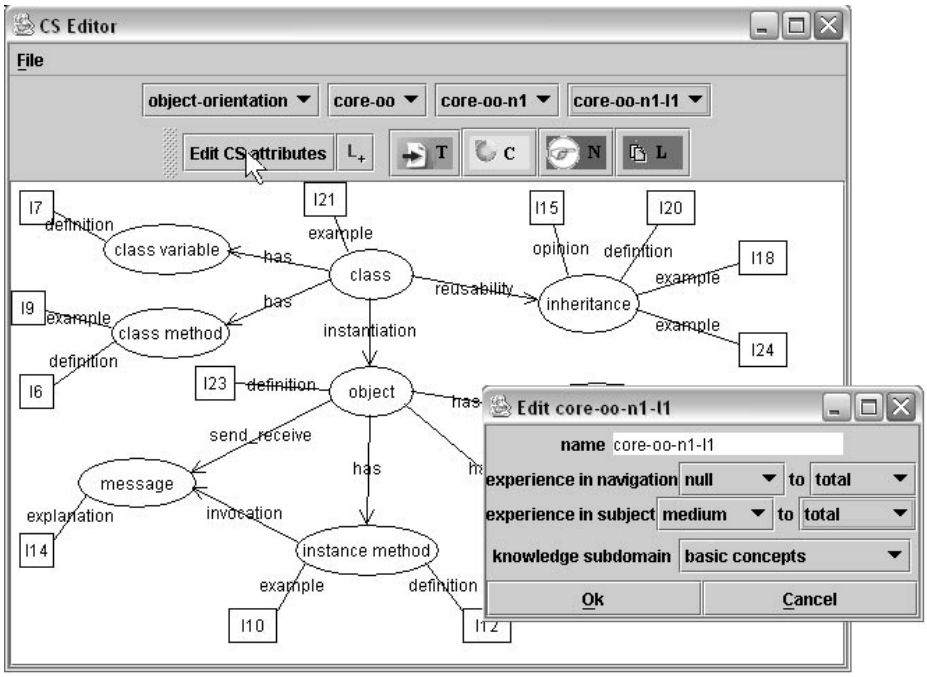


Figura 49: Etiquetado de la nueva EC_L

Paso III: Definir reglas de conocimiento para un ítem pulsando el botón derecho sobre éste y seleccionando la opción *Edit kowledge rules*.

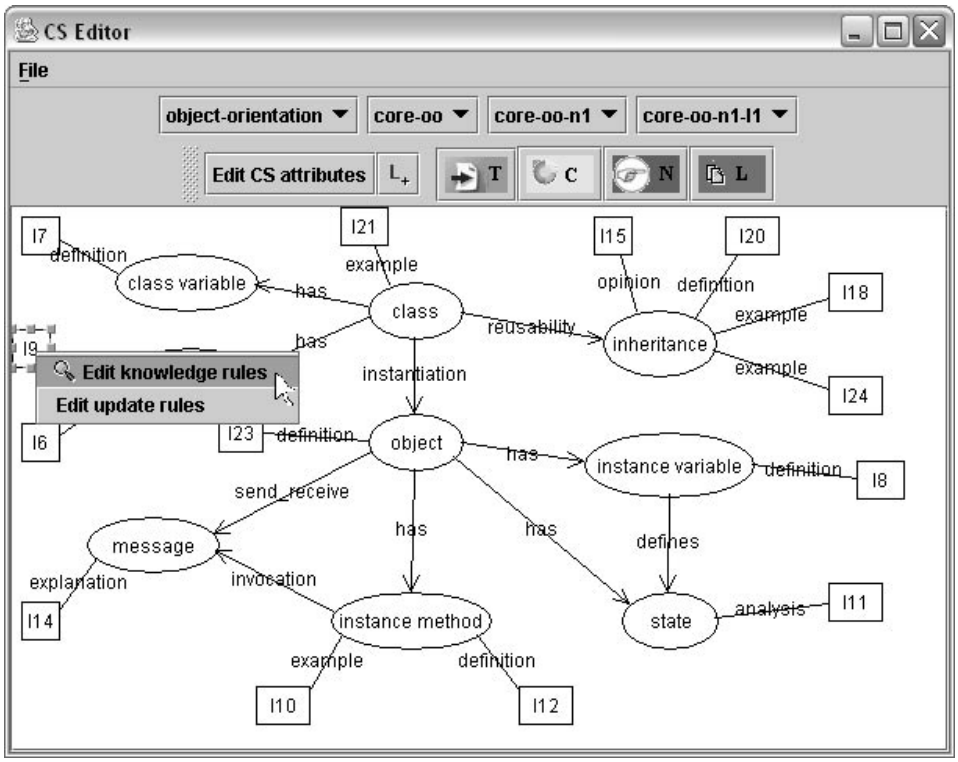


Figura 50: Editar las reglas de conocimiento asociadas a I9

Paso IV: Por defecto no existen reglas de conocimiento asociadas a un ítem, para añadir una nueva regla pulsar el botón *Add knowledge rule*.

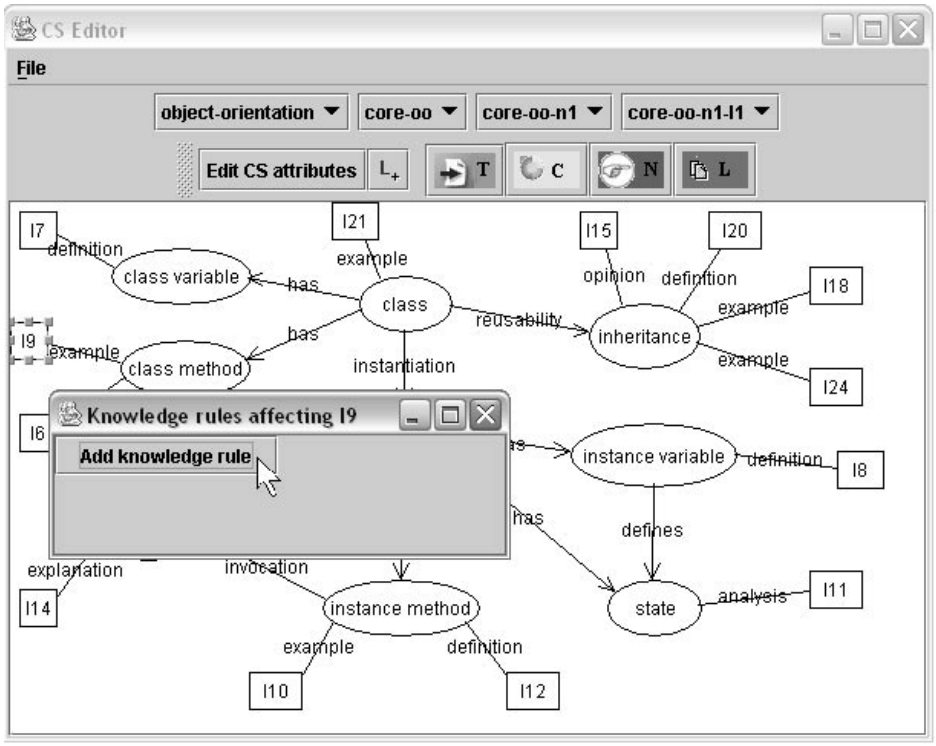


Figura 51: Añadir una regla de conocimiento para I9

Paso V: Para añadir prerrequisitos de conocimiento al cuerpo vacío de la regla pulsar botón derecho y seleccionar *add* en el menú que aparece. Después elegir el tipo de prerrequisito que se desea.

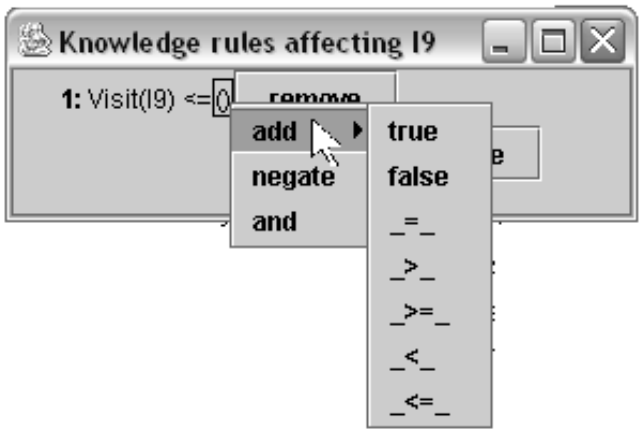


Figura 52: Añadir un predicado de conocimiento en Rk^1(I9)

Paso VI: Aparece un cuadro para dar valor al predicado. Para establecer el término izquierdo y derecho del predicado basta elegir una opción de la lista que aparece al situarnos encima y pulsar el botón derecho. Al terminar pulsar *Add* en el cuadro inicial.

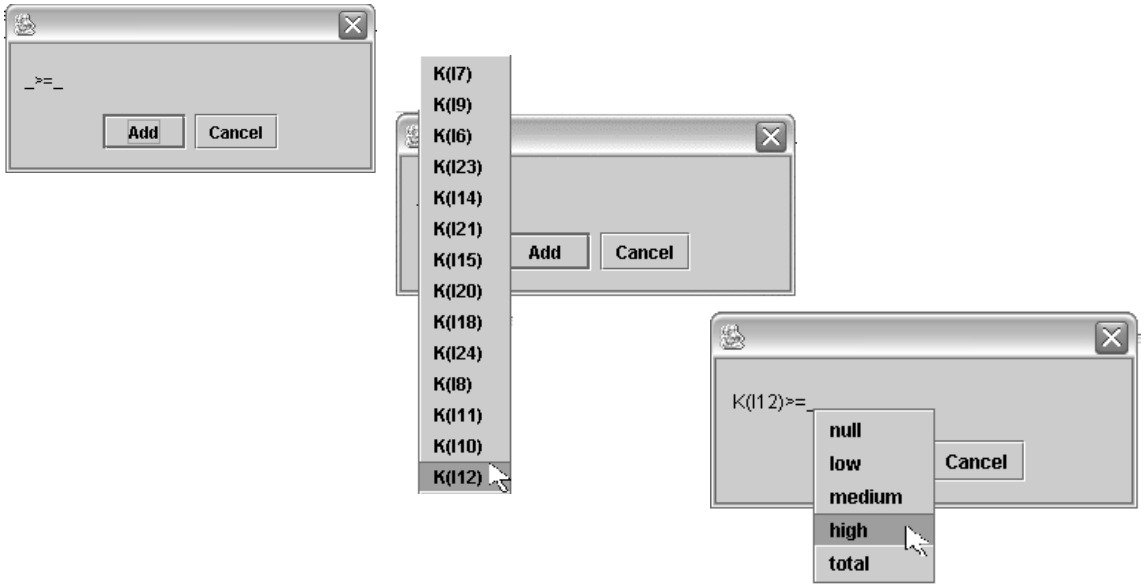


Figura 53: Crear el predicado *K(I12) >= high*

Una vez creado el predicado, se puede modificar cualquiera de sus términos pulsando el botón derecho sobre éste y eligiendo un nuevo valor de la lista. También se puede cambiar el tipo del predicado pulsando botón derecho sobre él y eligiendo la opción *replace with* con el predicado deseado.

Paso VII: Añadir más prerrequisitos de conocimiento en el cuerpo de la regla. Para ello, situarse sobre el predicado de conocimiento y pulsar botón derecho. Después elegir la opción *add at left* o *add at right* según se desee añadir el nuevo predicado a la izquierda o a la derecha del predicado actual.

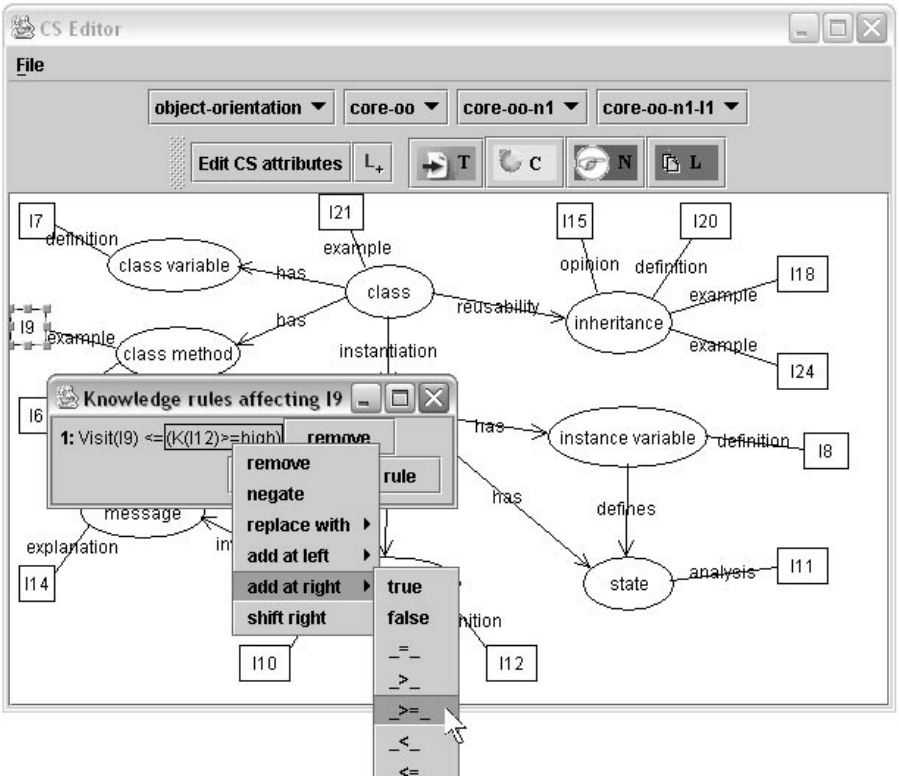


Figura 54: Añadir un predicado de tipo >= a la derecha del predicado actual

Paso VIII: Definir completamente el cuerpo de la regla y reordenar la ubicación de los predicados si se desea. Para mover un predicado, seleccionarlo y pulsar la opción *shift right* o *shift left* según se quiera desplazar el predicado a la derecha o a la izquierda. Si el predicado está en un extremo sólo se permite *shift right* (si es el primero) o *shift left* (si es el último).

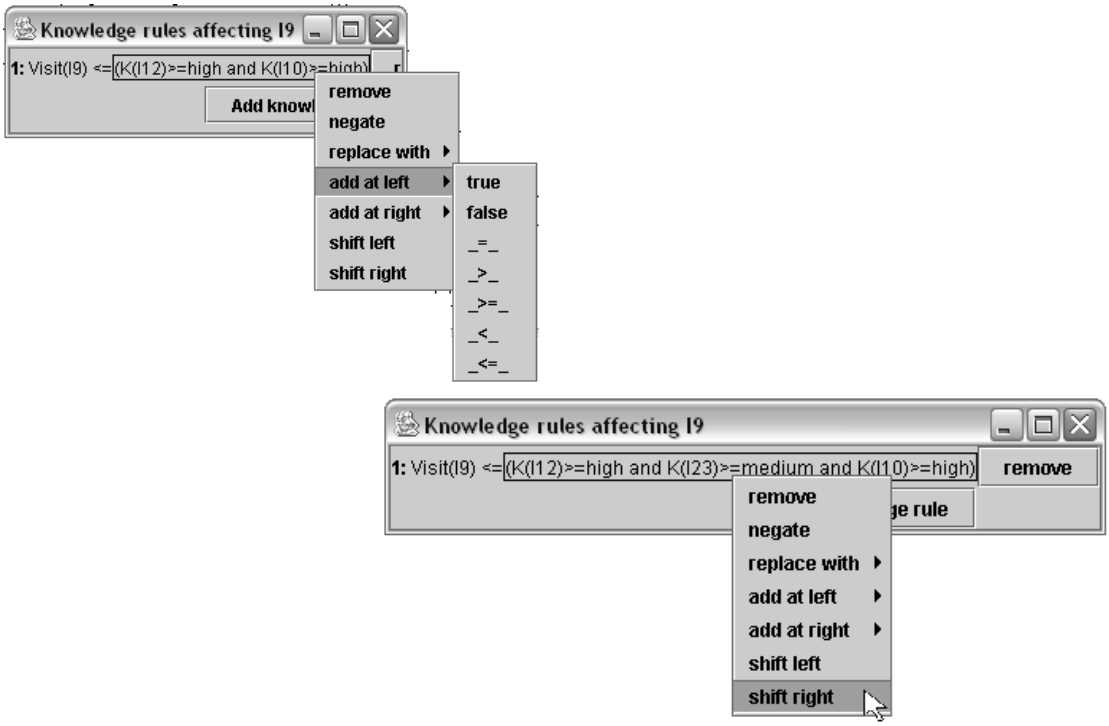


Figura 55: Añadir un predicado *K(I23) >= medium* en el centro y luego desplazarlo al final

Paso IX: Para definir más reglas de conocimiento asociadas al mismo ítem pulsar *Add Knowledge rule* y crear la nueva regla como se ha descrito en los pasos anteriores.

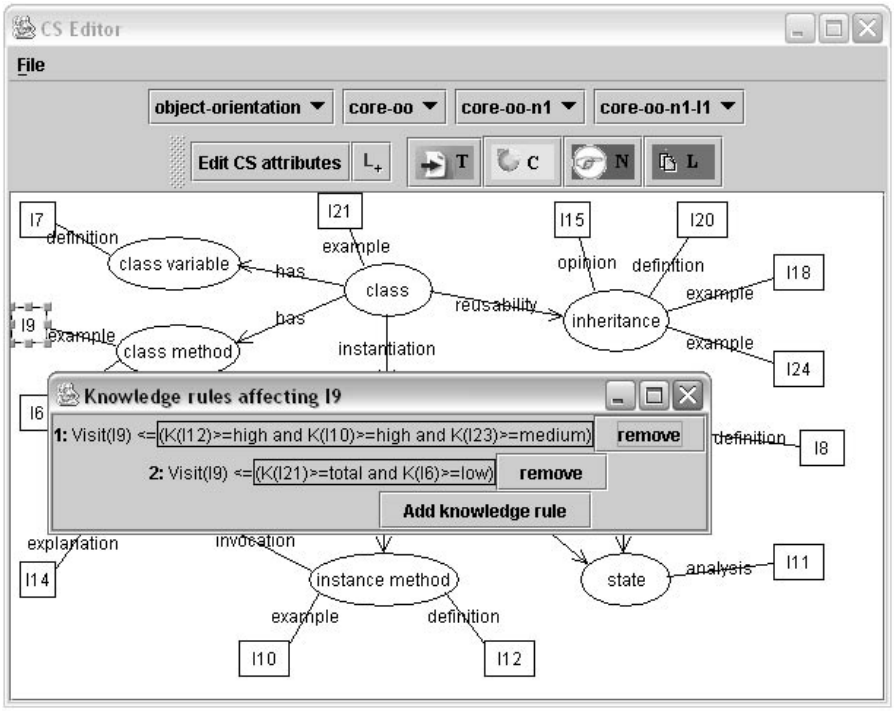


Figura 56: Dos reglas de conocimiento para I9

Para eliminar un predicado en el cuerpo de una regla se pulsa el botón derecho sobre él y se selecciona la opción *remove* en el menú que aparece. Para borrar toda la regla de conocimiento, pulsar el botón *remove* que aparece junto a ésta.

Paso X: Definir reglas de conocimiento para todos aquellos ítems que se desee. Por ejemplo, las que se muestran en las figuras 57 y 58.

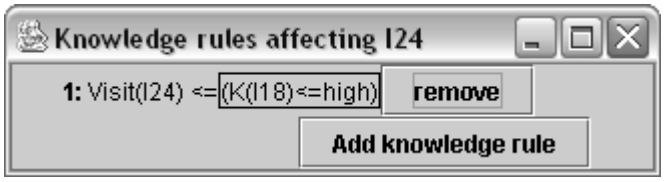


Figura 57: Regla de conocimiento para I24

La regla de I24 sólo impone un requisito de idoneidad para su visita: que el conocimiento sobre I18 no sea total.

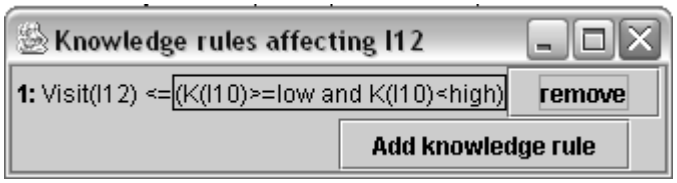


Figura 58: Regla de conocimiento para I12

La regla de I12 impone un requisito de accesibilidad y otro de idoneidad, ambos definidos sobre el ítem I10.

Paso XI: Editar la regla de actualización creada por defecto para un ítem, pulsando botón derecho sobre él y seleccionando la opción *Edit update rule*.

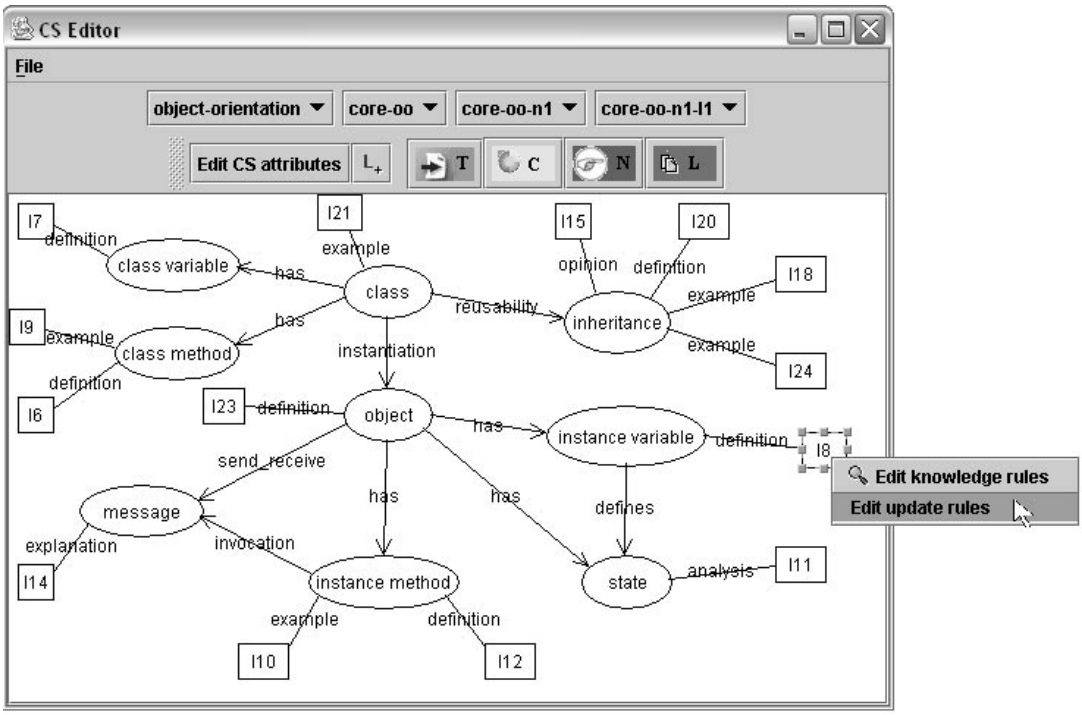


Figura 59: Editar la regla de actualización de I8

Paso XII: Para modificar el tipo de un predicado de actualización, situarse sobre él, pulsar botón derecho y seleccionar *replace with*. Cuando se trata del predicado que actualiza el ítem cabeza sólo son posibles dos tipos de predicados: *Fix-abs(_, total)* y *Inc-abs(_, _)*.

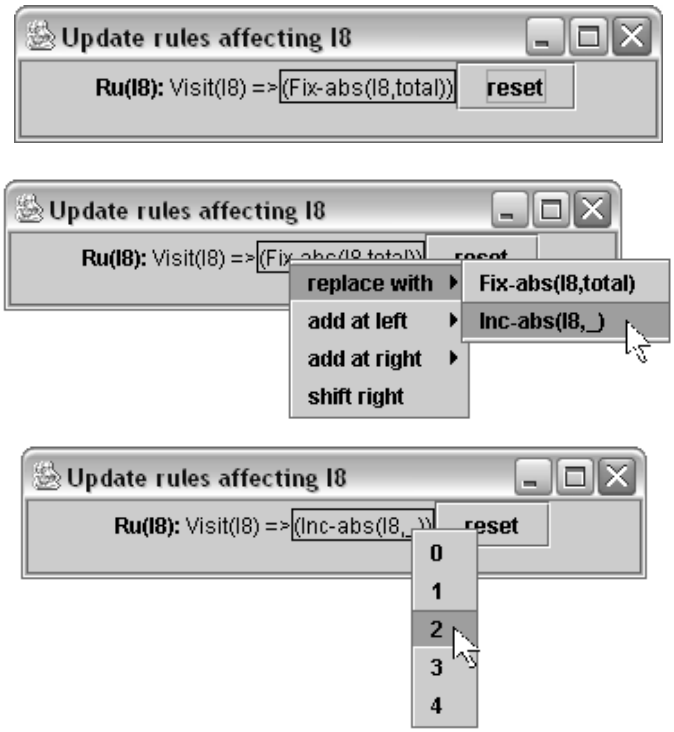


Figura 60: Regla de actualización para I8

Paso XIII: Modificar todas las reglas de actualización que se desee. Para añadir más predicados en el cuerpo de una regla usar las opciones *add at right* y *add at left* que aparecen al pulsar botón derecho sobre un predicado ya existente. Para establecer el valor de un argumento en un predicado pulsar botón derecho y elegir una opción de la lista (sólo se presentan las opciones válidas para ese tipo de predicado).

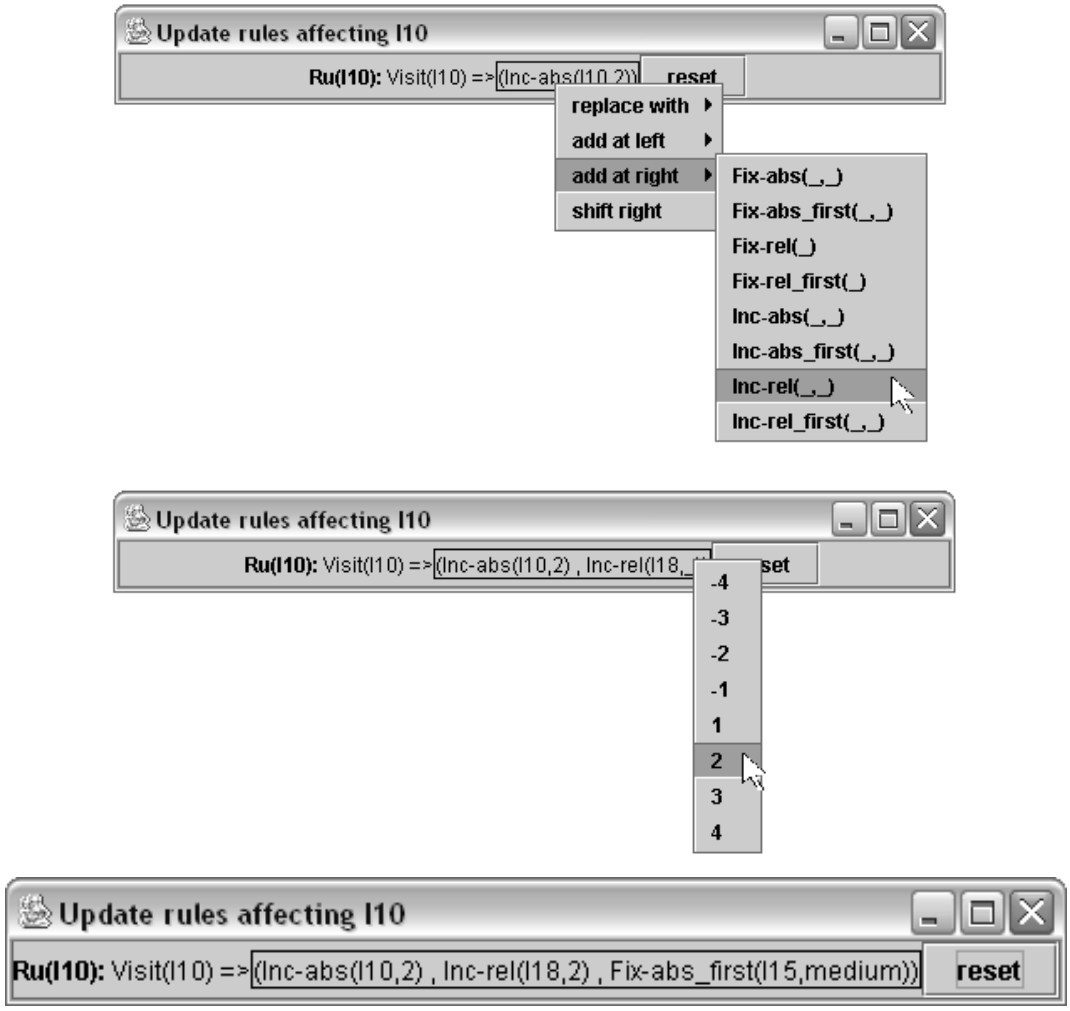


Figura 61: Regla de actualización para I10

En cualquier momento se puede utilizar el botón *reset* para cambiar la regla actual por la regla de actualización generada por defecto.

5.2 Navegación del sistema (usuario)

En la interfaz de autor existe una barra de navegación compuesta por cuatro botones, uno para cada modo de navegación posible: tradicional (T), por conceptos (C), por relación conceptual (N) y por conocimiento (L). De esta forma el autor se convierte en usuario y puede navegar la estructura actual para comprobar si funcionan correctamente las reglas definidas.

Figura 62: Barra de navegación

Paso I: Seleccionar el modo de navegación deseado pulsando sobre el icono correspondiente. Al hacerlo aparece una nueva ventana (*SEM-HP browser*), que permite navegar la estructura en el modo elegido.

En el ejemplo se ha seleccionado el modo de navegación por conocimiento. Actualmente sólo funcionan éste y la navegación tradicional.

Puesto que el conocimiento de partida del usuario es nulo, los ítems I9 e I12 aparecen ocultos y deshabilitados por no cumplirse las restricciones de accesibilidad establecidas para ellos en sus reglas de conocimiento, figuras 56 y 58 respectivamente.

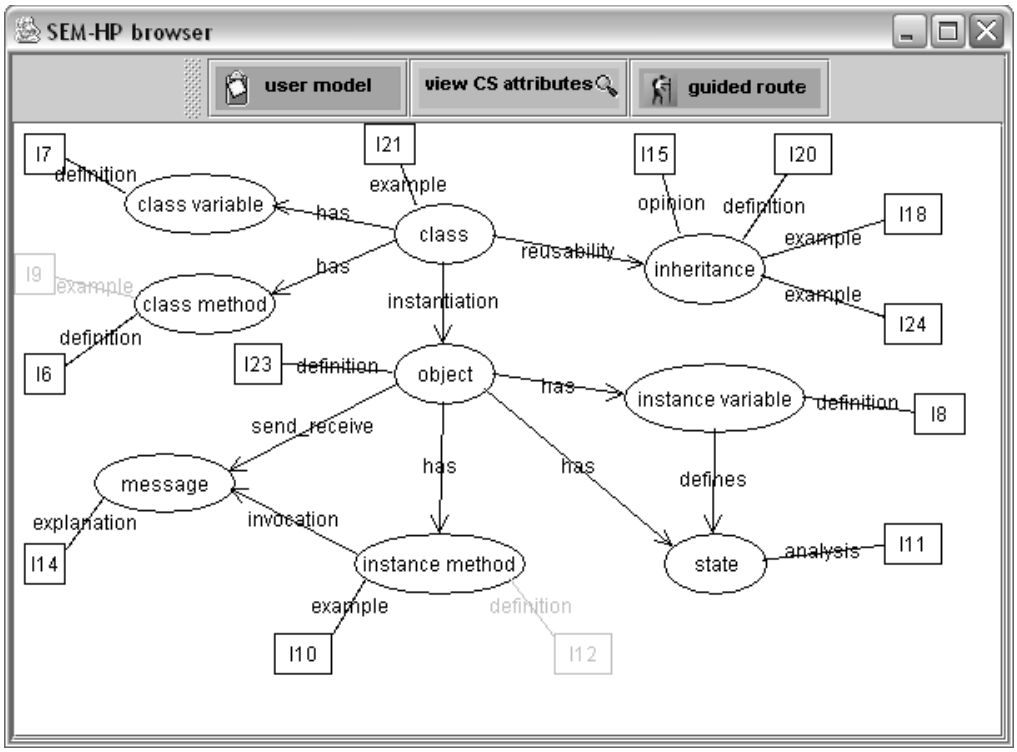


Figura 63: Navegación por conocimiento

Paso II: Si el usuario desea ver las propiedades de la estructura elegida puede pulsar el botón *view CS attributes* del menú superior.

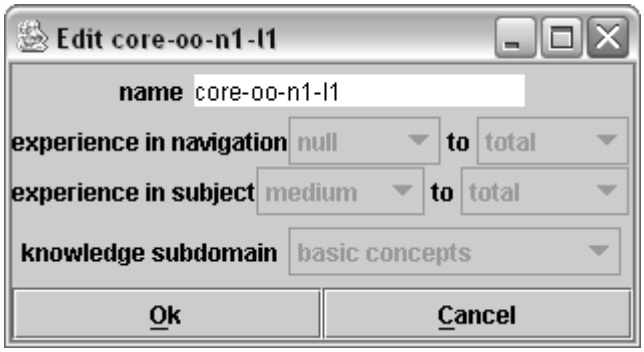


Figura 64: Etiquetado de la estructura de navegación

Observe que el etiquetado de la estructura se muestra, pero no se permite modificar (la funcionalidad de las listas desplegables se ha desactivado). Para hacer cambios es necesario trabajar en modo autor (ver figura 49).

Paso III: Si el usuario desea ver la información contenida en su modelo de usuario puede pulsar el botón *user model* situado en el menú superior. El modelo de usuario tiene dos pestañas.

La primera pestaña muestra para cada ítem el número de visitas y el grado de conocimiento del usuario. Además, permite al usuario marcar un ítem como interesante o definir una meta de conocimiento.

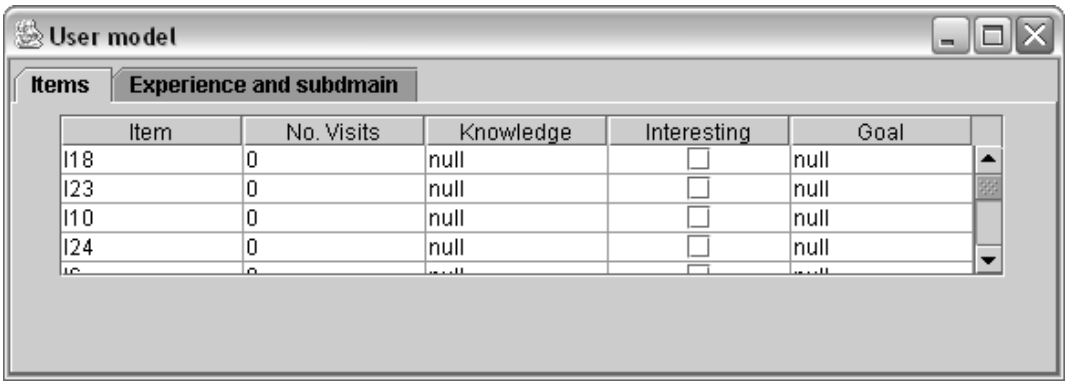


Figura 65: El modelo de usuario I

La segunda pestaña contiene el subdominio que interesa al usuario, así como su grado de experiencia de navegación y en la materia. Esta información puede ser modificada directamente por el usuario, eligiendo otro valor en las listas desplegables que se le proporcionan.

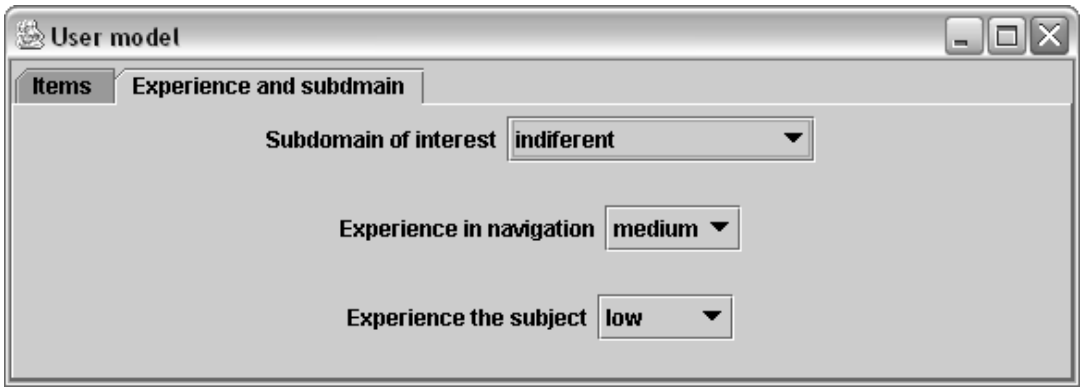


Figura 66: El modelo de usuario II

Paso IV: Para navegar, el usuario sólo tiene que hacer un *clic* sobre el ítem que desea visitar. Si éste es accesible, inmediatamente la información asociada al ítem se muestra en una ventana del navegador. En *Windows* se abre el navegador establecido por defecto, y en *Linux* se abre *Netscape*.

Acto seguido, el sistema ejecuta la regla de actualización asociada al ítem visitado y actualiza el modelo de usuario.

El grado de conocimiento que posee el usuario sobre cada ítem es anotado sobre la estructura de navegación mediante un sistema de colores. Este sistema consiste en rellenar el rectángulo que representa el ítem de un color más oscuro cuanto más lo conoce el usuario. Así, cuando el conocimiento del usuario sobre un ítem es “null”, el

rectángulo se rellena en blanco y si es “total” se rellena en negro. Se usan tres tonalidades de gris para los grados intermedios: “low”, “medium” y “high”.

En el ejemplo de la figura 67, se ha seleccionado el ítem I21, cuyas propiedades se editan en la figura 68. Observe que tras ejecutar la regla de actualización por defecto Ru(I21), el conocimiento del usuario sobre I21 es “total” (aparece en negro).

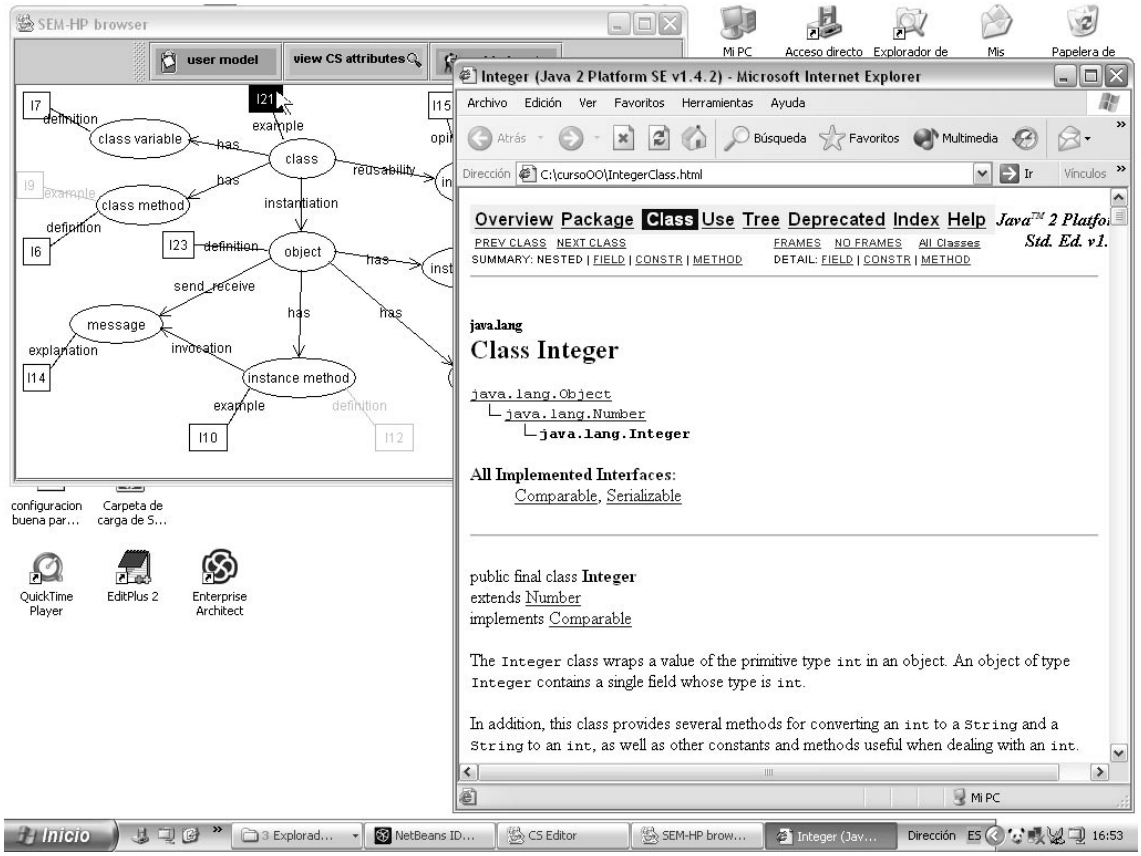


Figura 67: El usuario ha seleccionado I21

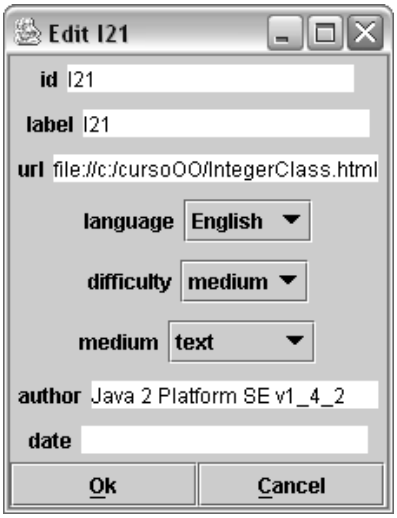


Figura 68: Propiedades del ítem I21

En este caso el documento mostrado se encuentra en el PC desde donde se ejecuta el prototipo (protocolo de conexión *file*). Por supuesto, también puede tratarse de un documento disponible a través de Internet (protocolo de conexión *http*).

Paso V: El usuario puede seguir visitando los ítems que desee, siempre que estos sean accesibles. Estas visitas harán accesibles ítems que ahora no lo son.

La figura 69 muestra el estado del *browser* después de que el usuario ha visitado una vez el ítem I10. Su regla de actualización, en la figura 61, hace que: el conocimiento del usuario sobre I10 se incremente dos niveles (de "null" a "medium"), el conocimiento sobre I18 se incremente dos niveles relativos a I10 (de "medium" a "total"), y el conocimiento sobre I15 se fije a "medium".

Con el nuevo estado, el ítem I12 se vuelve accesible por que el conocimiento sobre I10 supera el mínimo establecido en la regla Rk^1(I12) (figura 58). Y, el ítem accesible I24 deja de ser idóneo, porque el conocimiento sobre I18 ya no es inferior al conocimiento máximo establecido en la regla Rk^1(I24) (figura 57). La no idoneidad de I24 se refleja con una línea discontinua dibujando la asociación funcional y el borde del ítem.

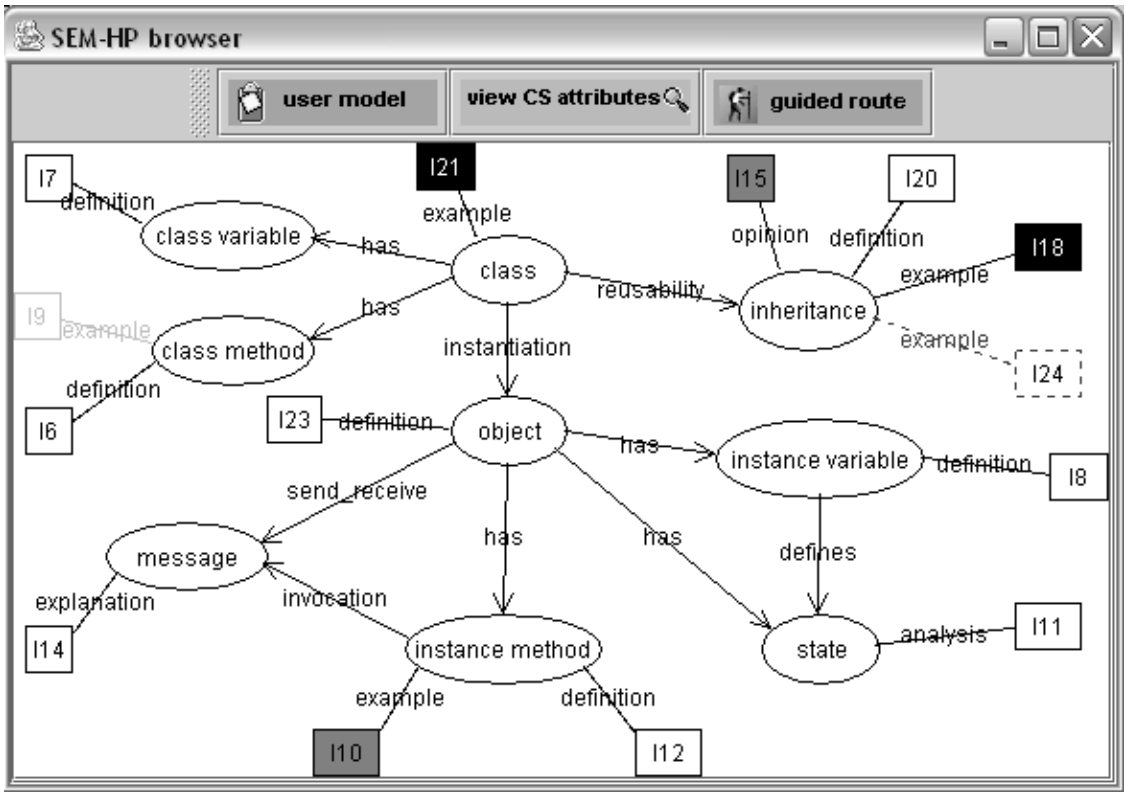


Figura 69: El usuario visita una vez I10

En la figura 70 se muestra la anotación realizada, después de que el usuario seleccione a partir de la figura 69 los ítems I10 e I6.

Al ejecutar, por segunda vez, la regla Ru(I10), el conocimiento sobre I10 llega a "total", por lo que el relleno de su rectángulo aparece negro y no ya en gris. Este aumento de conocimiento hace que el ítem I12 deje de ser idóneo, de acuerdo a lo establecido en la regla Rk^1(I12) (figura 58).

Por su parte, la visita de I6 fija a "total" el conocimiento del usuario sobre dicho ítem. Tras este cambio, se satisfacen todas las restricciones impuestas en la segunda regla de conocimiento de I9, Rk^2(I9) (figura 56), por lo que éste aparece como accesible e idóneo.

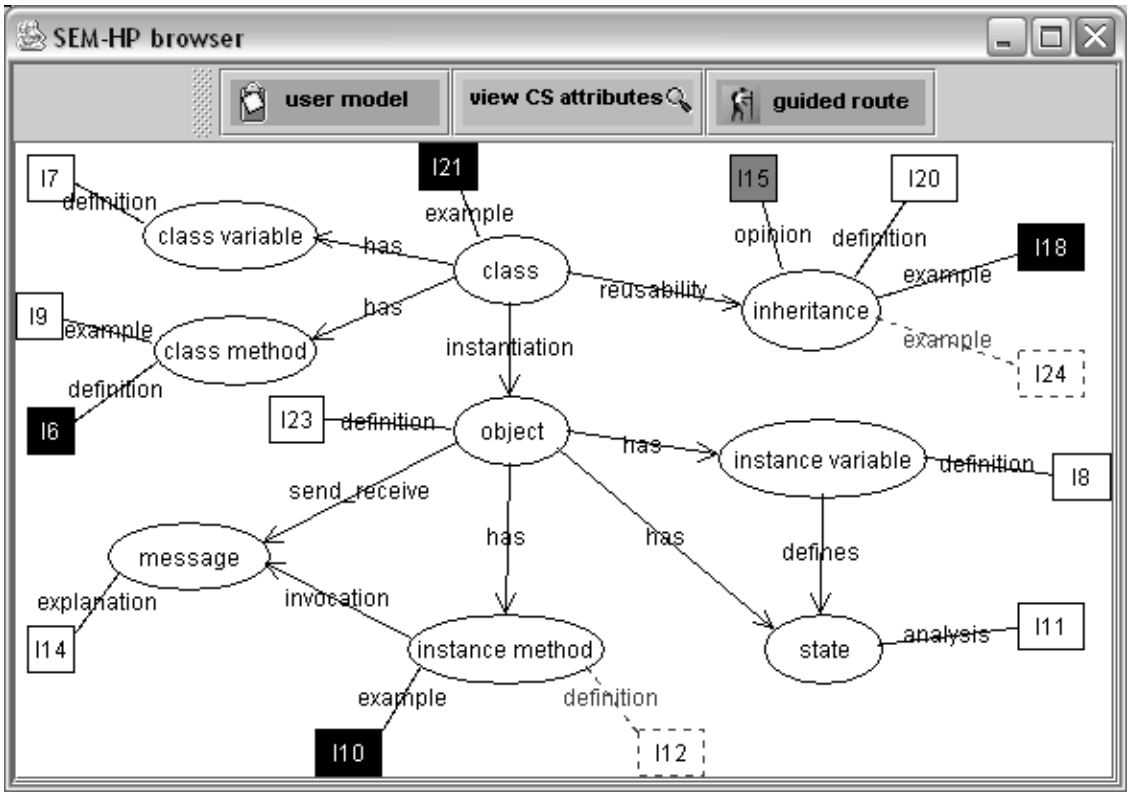


Figura 70: El usuario visita I10 e I6

Paso VI: Durante su navegación, el usuario puede marcar un ítem como interesante, caso de I20 en el modelo de usuario que se muestra en la figura 71. Si el ítem es accesible se marca automáticamente como deseable en la estructura de navegación, utilizando un borde más grueso y de color rojo (ver la figura 72).

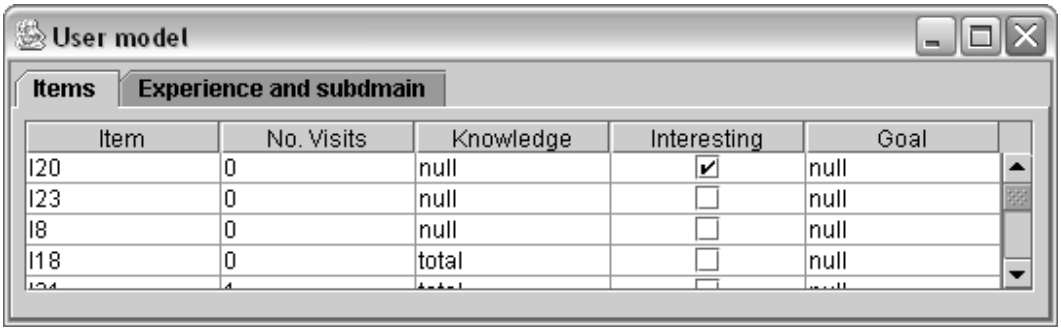


Figura 71: I20 marcado como interesante en el modelo de usuario

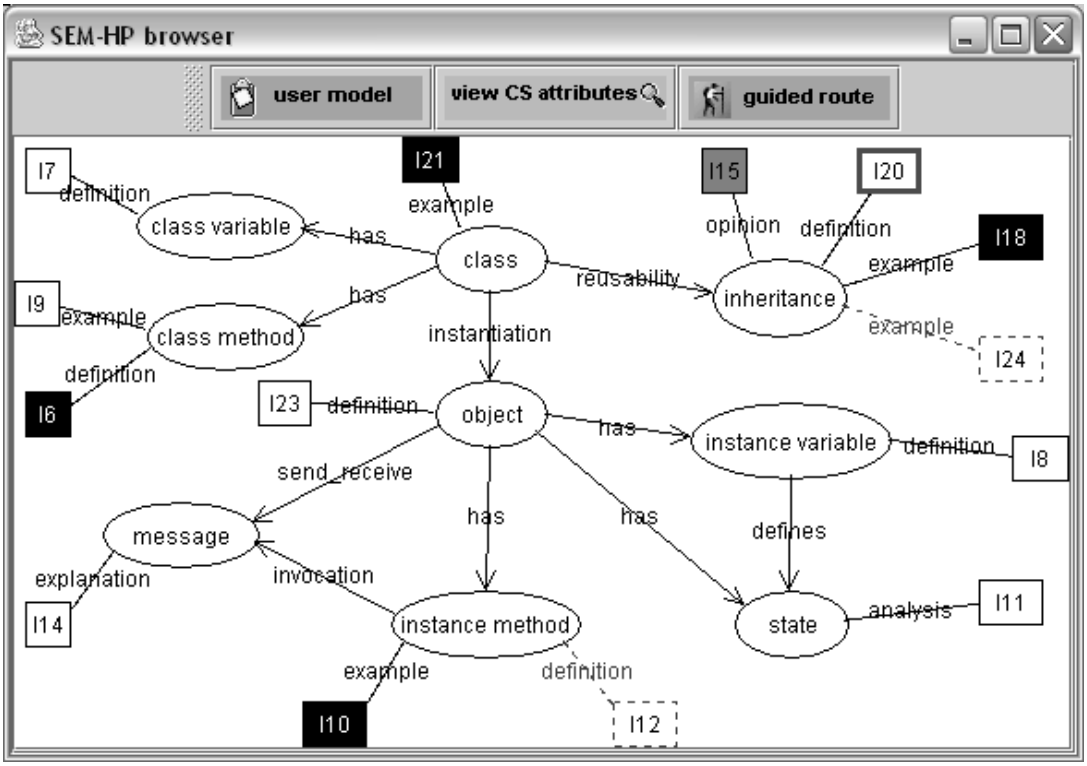
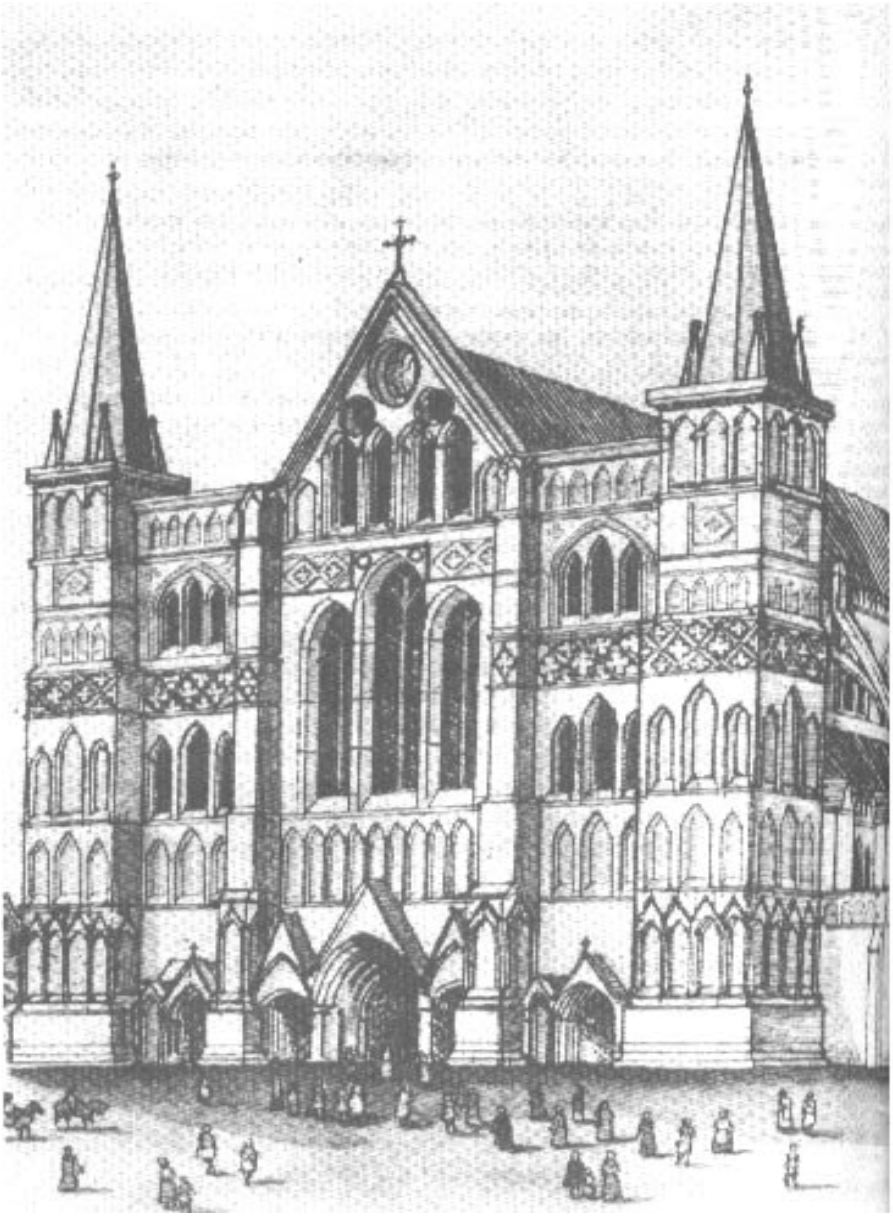


Figura 72: I20 anotado como deseable en la estructura de navegación

Módulo VI

Conclusiones



Los Pilares de la Tierra

Módulo VI

CAPÍTULO 23

Consideraciones sobre Evolución y Adaptación

Resumen

El objetivo de este capítulo es caracterizar el modelo SEM-HP desde los puntos de vista de la evolución y la adaptación al usuario. Para lo primero, se describen los diferentes modelos que un sistema software puede seguir para evolucionar y se justifican cuáles de estos modelos están presentes en SEM-HP, haciendo especial hincapié en la capacidad evolutiva del Sistema de Aprendizaje. Para lo segundo, se califica la adaptación realizada en SEM-HP respecto a los distintos criterios que se establecen en la taxonomía de sistemas hipermedia adaptativos publicada en [Medina, 02c]. Durante este capítulo se argumenta, también, que evolución y adaptación al usuario son dos conceptos distintos, pero que su intersección no es vacía. Concluyendo finalmente que la adaptación realizada en los sistemas hipermedia adaptativos es sólo un caso concreto de evolución.

Tabla de contenidos

Consideraciones sobre Evolución y Adaptación

1. INTRODUCCIÓN

En el proceso evolutivo de un sistema software el desarrollador es un elemento muy importante, ya que es el encargado de modelar y diseñar la capacidad evolutiva del sistema y de, posteriormente, hacer uso de esas acciones evolutivas para realizar los cambios necesarios.

Sin embargo, no es el único elemento que debe ser considerado. La figura 1 muestra una estructura de interacción abstracta que representa los principales elementos en la evolución de un sistema software [Parets, 95] [Rodríguez, 01]:

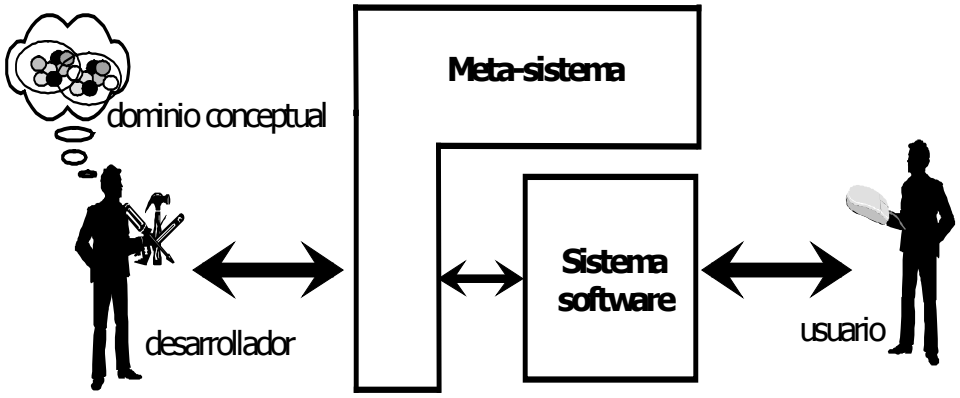


Figura 1: Estructura de interacción

- el **desarrollador** decide las modificaciones que van a poder ser realizadas sobre el sistema software,
- el **meta-sistema** permite la interacción entre el sistema software y el desarrollador,
- el **sistema software** sufre los cambios, y
- finalmente, el **usuario** utiliza el sistema.

Desde una perspectiva general son dos las formas en que puede evolucionar un sistema software:

a) Evolución dirigida por el desarrollador.

b) Evolución dirigida por el propio sistema.

Aunque el desarrollador interviene en ambas, la diferencia entre ellas radica en si esta intervención es directa o indirecta. El primer tipo de evolución **(a)** implica una **mediación directa del desarrollador**, pues, es éste quien dirige los cambios en el sistema software. El segundo tipo **(b)**, se realiza de forma **automática** dependiendo de

ciertos mecanismos definidos previamente por el desarrollador, con lo cual su participación en este segundo caso es indirecta.

La *adaptación al usuario* que se lleva a cabo generalmente en los sistemas hipermedia adaptativos, se realiza de forma *automática* mientras que el usuario navega; por lo tanto el modelador no interviene de forma directa en la adaptación. De hecho, en la mayoría de los casos, ni siquiera el usuario solicita explícitamente la adaptación, y de ser así, el sistema hipermedia se considera adaptable en lugar (o además) de adaptativo.

Por lo tanto, **la adaptación** realizada en este tipo de sistemas **entra dentro de la segunda forma de evolución** (b). Los mecanismos definidos previamente por el desarrollador para conseguir que el sistema se adapte por sí mismo, son habitualmente: el modelo de usuario y un conjunto de métodos de adaptación destinados a personalizar la presentación de la información y la estructura de enlaces.

También forman parte de esos mecanismos los elementos internos que permiten y dirigen la aplicación de los métodos de adaptación, así como la actualización del modelo de usuario. En nuestro caso, se trataría de las reglas de orden, de conocimiento, de peso y de actualización.

Con objeto de enmarcar, de manera más precisa, la adaptación al usuario realizada en los sistemas hipermedia adaptativos en el campo de la evolución, se describen en la siguiente sección, los mecanismos y modelos de evolución propuestos por Torres-Carbonell en su tesis doctoral [Torres, 02].

2. MECANISMOS DE EVOLUCIÓN

Los mecanismos de evolución representan las distintas *formas usadas por un sistema software para cambiar*. Cada mecanismo incluye un conjunto de actividades, tales que su ejecución coordinada produce el cambio. En [Torres, 99] se proponen dos tipos de mecanismos: herencia y adaptación.

2.1 Herencia

El mecanismo de la herencia es usado para producir **sistemas software descendientes** que hereden las adaptaciones realizadas por sus padres. Es decir, el sistema hijo hereda la estructura inicial y los cambios realizados sobre ésta como consecuencia de los dos tipos de adaptación que se explican en la sección posterior.

2.2 Adaptación

Igual que ocurre en biología, la adaptación se basa en la necesidad de **acomodarse, aprender o mutar** de acuerdo a los requerimientos del entorno. Una adaptación puede implicar cambios de estructura o cambios de funcionamiento en un sistema software. Según el cambio que tiene lugar en la adaptación, se distinguen dos tipos (resumidos en la tabla 1):

- **Adaptación por Acomodación/Aprendizaje**: El sistema se adapta a su entorno, *aprendiendo el mejor modo de usar su estructura sin modificarla*. Este tipo de adaptación tiene lugar en entornos funcionales, donde el usuario se comunica con el sistema a través de las acciones de su interfaz y percibe la adaptación como cambios en las respuestas de éste.

- **Adaptación por Mutación/Diferenciación**: Este tipo de adaptación es más radical que el anterior, ya que *implica cambios en la estructura del sistema software*. Las modificaciones estructurales provocan cambios en la funcionalidad del sistema, pero además introducen nuevas posibilidades de adaptación por acomodación/aprendizaje. El sistema software no es capaz de realizar cambios en su estructura por sí solo, para ello requiere la intervención directa del desarrollador y del meta-sistema.

Tabla 1: Mecanismos de adaptación

ADAPTACIÓN	DESCRIPCIÓN	CAMBIOS ESTRUCTURALES	CAMBIOS FUNCIONALES
Acomodación/Aprendizaje	Tiene lugar en entornos funcionales	No	Sí
Mutación/Diferenciación	Introduce nuevas posibilidades de adaptación por acomodación/aprendizaje	Sí	Sí

El entorno de un **sistema hipermedia adaptativo** es un **entorno funcional**, debido a que el usuario navega, accede y lee hiperdocumentos haciendo uso de la interfaz de navegación que el sistema le ofrece. Además, el sistema se adapta al usuario cambiando la forma de usar su estructura pero sin modificarla.

Las técnicas de adaptación de la navegación consisten en ocultar, anotar, deshabilitar o reordenar los enlaces mientras que las técnicas de adaptación de la presentación muestran u ocultan fragmentos. Como puede deducirse a partir de éstas, los métodos de adaptación de los sistemas hipermedia adaptativos [Brusilovsky, 96] **no implican cambios estructurales**.

Además, ninguno de los sistemas hipermedia adaptativos revisados en la literatura incluye el concepto de meta-sistema ni nada conceptualmente similar que permita realizar modificaciones estructurales. Como resultado de este análisis, se concluye que la *adaptación al usuario realizada en los sistemas hipermedia adaptativos se puede catalogar dentro de los mecanismos de adaptación por acomodación/aprendizaje*.

3. MODELOS DE EVOLUCIÓN

Siguiendo [Torres, 99] un modelo de evolución es una *representación simbólica de un forma particular de introducir cambios en un sistema software*. Cada modelo utiliza

alguno de los mecanismos evolutivos descritos en la sección anterior. Concretamente, se consideran seis modelos diferentes de evolución:

1. **Meta-teológico dirigido por el modelador**.

El modelador realiza modificaciones en la estructura del meta-sistema usando las acciones disponibles en la interfaz de evolución del mismo. Este modelo aplica el mecanismo de adaptación por mutación/diferenciación.

2. **Teológico dirigido por el modelador**.

El modelador, a través de las acciones de interfaz del meta-sistema, produce cambios en la estructura del sistema software. Aplica el mecanismo de adaptación por mutación/diferenciación.

3. **Herencia de los meta-caracteres adquiridos**.

Por iniciativa del modelador, a partir de un meta-sistema existente (padre), se genera un nuevo meta-sistema (hijo), que hereda los meta-caracteres adquiridos por el padre durante su evolución. Este modelo aplica el mecanismo de herencia.

4. **Herencia de los caracteres adquiridos**.

Este modelo es seguido por el modelador cuando desea crear un nuevo sistema software a partir de uno ya existente. El sistema generado adopta (utilizando el mecanismo de herencia) las nuevas características que el viejo sistema había adquirido durante su evolución.

5. **Adaptación del meta-sistema por sí mismo**.

El meta-sistema realiza cambios estructurales en el sistema software, sin intervención directa del modelador. Este modelo aplica el mecanismo de adaptación por mutación/diferenciación.

6. **Adaptación del sistema software por sí mismo**.

El sistema realiza un proceso adaptativo sin intervención directa del modelador, ni del meta-sistema. Este modelo aplica el mecanismo de adaptación por acomodación/aprendizaje.

La tabla 2 confronta las características de los seis modelos descritos. La figura 2 presenta de forma gráfica esta misma información. Y en la figura 3 se superpone cada uno de los seis modelos sobre la estructura de interacción de la figura 1 (cada flecha lleva asociado el número del modelo que representa).

Tabla 2: Modelos de evolución

CAMBIOS	DIRIGIDOS POR	A TRAVÉS DE	SUFRIDOS POR	MECANISMO
MODELO 1.- Meta-Teleología dirigida por el Modelador				
Estructurales	Modelador	Interfaz de evolución del meta-sistema	Meta-sistema	Mutación/ Diferenciación
MODELO 2.- Teleología dirigida por el Modelador				
Estructurales	Modelador	Interfaz de acción del meta-sistema	Sistema	Mutación/ Diferenciación
MODELO 3.- Herencia de los Meta-Caracteres adquiridos				
Meta-sistema hijo	Modelador	Interfaz de acción del meta-sistema	Meta-sistema	Herencia
MODELO 4.- Herencia de los Caracteres adquiridos				
Sistema hijo	Modelador	Interfaz de acción del meta-sistema	Sistema	Herencia
MODELO 5.- Autoadaptación del Meta-sistema				
Estructurales	Meta-sistema	Interfaz de acción del meta-sistema	Sistema	Mutación/ Diferenciación
MODELO 6.- Autoadaptación del Sistema Software				
Funcionales	Sistema	Mecanismos de adaptación	Sistema	Acomodación/ Aprendizaje

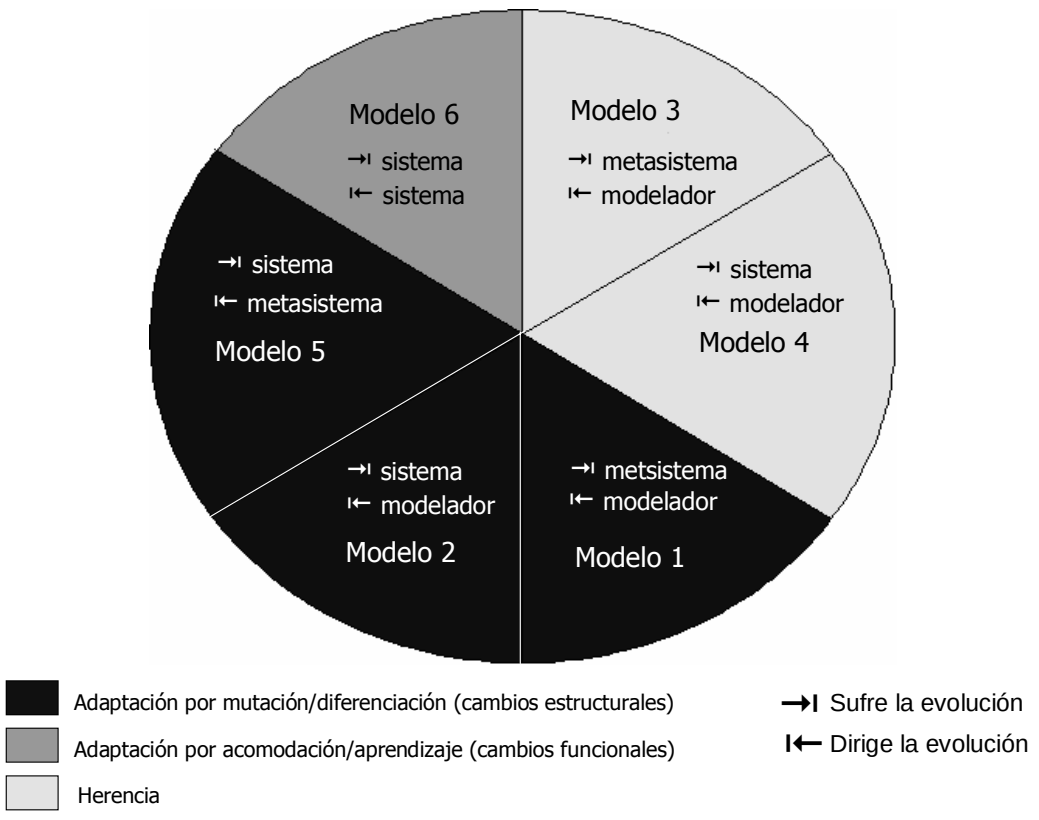


Figura 2: Modelos de evolución (i)

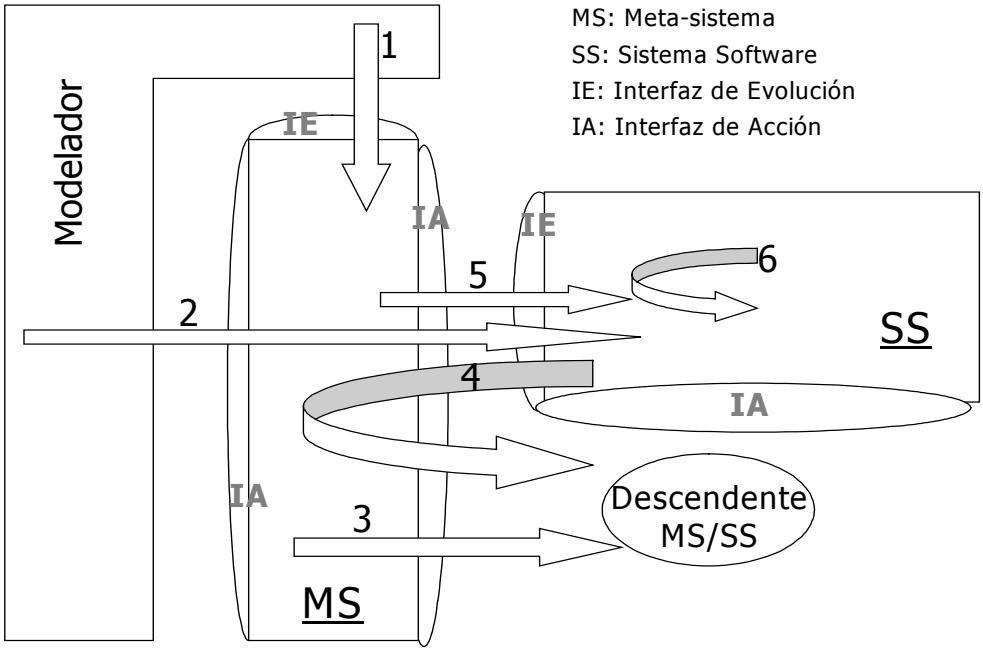


Figura 3: Modelos de evolución (ii)

Los **sistemas hipermedia adaptativos** realizan el proceso de adaptación al usuario mientras que éste navega en el sistema, sin intervención directa del modelador. Además, como se justificó en la sección 2.2, aplican mecanismos de adaptación por acomodación/aprendizaje. Por lo tanto, encajan perfectamente en el **sexto modelo de evolución** (Autoadaptación del sistema software).

El anterior estudio nos lleva a afirmar que la evolución del sistema hipermedia es un concepto mucho más amplio que la adaptación al usuario [Medina, 02b]. Sin embargo, no se trata de conceptos disjuntos, muy al contrario, existe una fuerte relación entre ambos que establece la adaptación al usuario como un importante modelo de evolución, pero no el único. Dicho de otro modo, la adaptación al usuario está incluida dentro del proceso de evolución del software que, además de éste, incluye otros cinco modelos para el cambio (véase la figura 4).

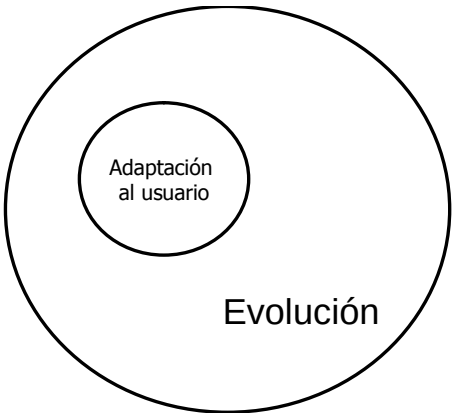


Figura 4: Adaptación al usuario $\subset$ Evolución del software

4. CARACTERIZACIÓN EVOLUTIVA DE SEM-HP

Hasta ahora, se han descrito, brevemente, las distintas formas en las que un sistema software puede evolucionar (tablas 1 y 2), poniendo de manifiesto que la adaptación realizada en la mayoría de los sistemas hipermedia adaptativos se limita al sexto modelo de evolución. En la presente sección se establece y justifica cómo los sistemas hipermedia desarrollados de acuerdo al modelo SEM-HP son capaces de proporcionar cuatro de los seis modelos de evolución estudiados [Medina, 02f].

4.1 Meta-Teleología dirigida por el modelador (modelo 1)

En SEM-HP, el primer modelo de evolución es soportado parcialmente. Esto se debe a que el único componente del meta-sistema que el autor puede modificar, usando la interfaz de evolución de éste, son las restricciones que el autor impone para la ejecución de las acciones evolutivas (RTa).

El autor puede añadir o cambiar las RTa pero en ningún caso las restricciones del sistema (RTs), las acciones evolutivas (ACe), los mecanismos de propagación del cambio o las meta-restricciones encargadas de asegurar la consistencia en la evolución de las RTa [García, 01c].

4.2 Teleología dirigida por el modelador (modelo 2)

En SEM-HP, el autor puede cambiar los distintos componentes que constituyen el sistema hipermedia usando las **acciones de evolución** disponibles a través de la interfaz de acción del meta-sistema. Cada una de estas modificaciones debe verificar una serie de restricciones, impuestas por el sistema y por el autor.

El meta-sistema se encarga de comprobar ambos tipos de restricciones, RTs y RTa, e informar al autor en caso de que no sea posible realizar de forma consistente la modificación requerida. Puesto que las ACe producen *cambios en la estructura del sistema* también generarán cambios en su funcionamiento. De esto modo, podemos incluir las ACe dentro de los mecanismos de adaptación por mutación/diferenciación.

En el Sistema de Aprendizaje, las acciones evolutivas permiten modificar la estructura de las reglas de peso (capítulo 11), de conocimiento (capítulo 8) y de actualización (capítulo 9), así como el esquema del modelo de usuario (capítulo 12) y el etiquetado de las estructuras de aprendizaje (capítulo 14). Estas modificaciones repercuten, obviamente, en la navegación que el usuario va a realizar durante su interacción con el sistema: desde la estructura de navegación que se le proporciona hasta la forma en que adquiere el conocimiento, pasando por la accesibilidad e idoneidad de los ítems ofrecidos.

4.3 Autoadaptación metasistema-sistema (modelo 5)

El autor puede efectuar cambios en el sistema usando las ACe que el meta-sistema le ofrece. Sin embargo, esto no es suficiente, ya que una modificación realizada en un

sistema puede tener algún tipo de repercusión sobre otros componentes de éste (**propagación interna del cambio**) o incluso de otros sistemas con los que se interrelaciona (**propagación externa del cambio**).

Para garantizar la consistencia global del sistema[Medina, 03d] tras las modificaciones que el autor realiza, el proceso de propagación del cambio es llevado a cabo, de forma automática, por el meta-sistema. Este proceso genera cambios estructurales, y por lo tanto, puede ser clasificado como un mecanismo de adaptación por mutación/diferenciación.

Los cambios que se propagan al Sistema de Aprendizaje son fundamentalmente aquellos que modifican las asociaciones funcionales existentes en las estructuras conceptuales de memorización y presentación, así como los que eliminan (u ocultan) y crean (o muestran) ítems en éstas.

Estos cambios suelen implicar la eliminación o modificación de las reglas de conocimiento, de actualización y de peso que contienen a los ítems involucrados. La propagación puede implicar, también, la creación de nuevas reglas de actualización o de peso cuando el cambio realizado consiste en la adicción de un ítem o un concepto en la estructura conceptual correspondiente (capítulo 13, Propagación externa del cambio).

Las figuras 5.1 y 5.2 muestran dos ejemplos de propagación del cambio que el meta-sistema realiza automáticamente como consecuencia de la ejecución de una acción evolutiva por parte del autor. Ambos ejemplos implican cambios en la estructura del sistema hipermedia, y por lo tanto, también en su funcionalidad.

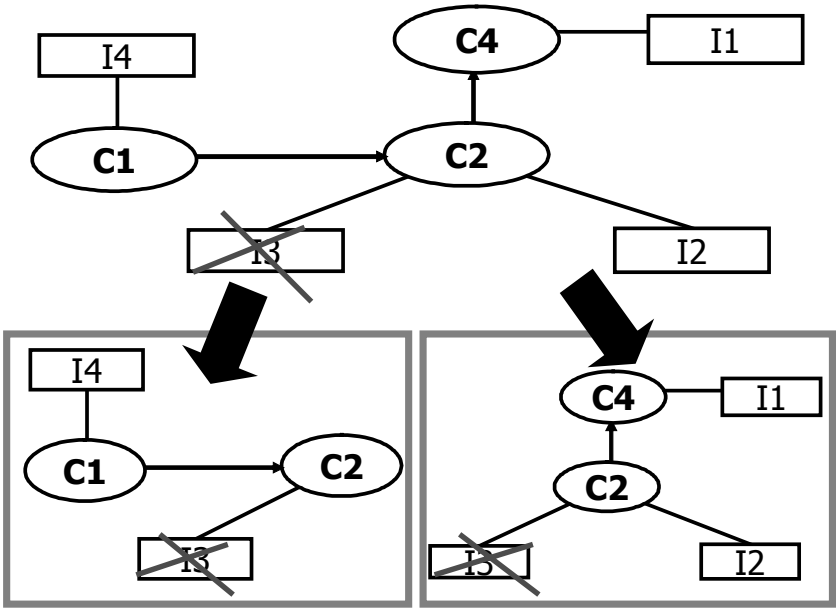
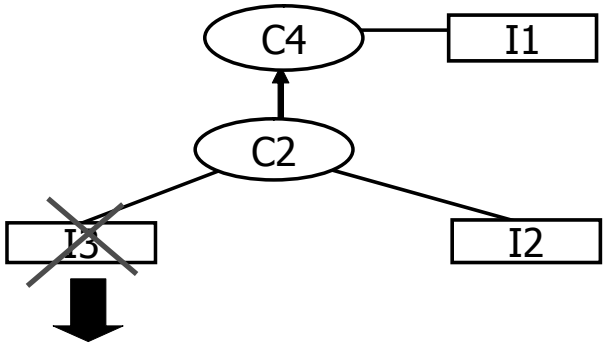


Figura 5.1: Propagación SM → SP

En la figura 5.1 se muestra cómo al eliminar el ítem I3 en la estructura conceptual del Sistema de Memorización (SM) se desencadena una propagación sobre el Sistema de

Presentación (SP), que supone la eliminación en cascada de dicho ítem en todas las estructuras conceptuales de presentación que lo incluyen.



- Regla de conocimiento:

~~Rk(I3): K(I1) ≥ "alto" and K(I2) ≥"medio" → Visitar(I3)~~

- Reglas de actualización:

Ru(I1): Si visitado(I1) entonces Fix-abs(I1, "total");

Ru(I2): Si visitado(I2) entonces Fix-abs(I2, "total"), ~~Inc-abs(I3, 2)~~;

~~Ru(I3): Si visitado(I3) entonces Fix-abs(I3, "total");~~

Figura 5.2: Propagación SP → SA

En la figura 5.2 se puede observar cómo al eliminar el ítem I3 en el Sistema de Presentación se requiere eliminar ese ítem en las reglas del Sistema de Aprendizaje (SA) que lo implican. Concretamente, en el ejemplo, se han eliminado las reglas de conocimiento y actualización asociadas a I3, y se ha modificado la regla de actualización para I2 en cuyo cuerpo aparecía un predicado definido sobre I3.

4.4 Autoadaptación del sistema hipermedia (modelo 6)

Los sistemas hipermedia adaptativos desarrollados de acuerdo al modelo SEM-HP realizan el proceso de adaptación al usuario de forma automática mientras que el usuario navega, sin intervención directa del modelador, ni del meta-sistema. Para ello, utilizan mecanismos previamente definidos por el modelador, como son el modelo de usuario y los métodos de adaptación. Concretamente los **métodos de adaptación** utilizados en SEM-HP son:

- **Consejo local**: Mediante la anotación de los *ítems deseables*, el sistema informa al usuario de aquellos ítems cuya visita, en el siguiente paso de navegación, le acerca, en mayor o menor grado, a la consecución de un estado de conocimiento que satisface una meta marcada o le permite visitar los ítems que le interesan.

- **Consejo global**: Por medio de las *rutas guiadas personalizadas*, el sistema conduce la navegación del usuario indicándole, en cada paso, el ítem que debe visitar a continuación. De tal modo que, una vez completada la ruta, el estado alcanzado por el usuario es seguro que satisface su meta de conocimiento.

- **Soporte a la orientación**: El sistema reduce la desorientación del usuario poniendo a disposición de éste una *estructura de navegación semántica*, donde en todo momento se visualiza el último ítem visitado, el concepto al que éste se asocia, otros ítems ligados a ese mismo concepto, y los conceptos relacionados. Además, dependiendo del modo de navegación, sobre la propia estructura se refleja *información acerca de si un ítem ha sido o no visitado antes, cuántas veces, e incluso, cuánto conocimiento se posee sobre él*.

- **Vistas personalizadas orientadas al conocimiento del usuario**: El sistema *oculta y deshabilita los ítems inaccesibles* con el objetivo de disuadir e impedir la visita de éstos. Los criterios que se siguen para identificar si un ítem es accesible o no dependen del usuario y del modo de navegación utilizado (capítulo 15).

- **Vistas personalizadas orientadas a la experiencia e intereses del usuario**: El sistema *elige de forma personalizada la estructura de navegación* proporcionada a cada usuario, comparando el etiquetado de las estructuras existentes con la información almacenada en el modelo de usuario acerca de su experiencia y subdominio de interés.

Estos mecanismos modifican la forma en que el usuario puede usar la estructura del sistema hipermedia, pero no la modifican.

Un ejemplo claro, es la deshabilitación de un ítem inaccesible. El ítem sigue existiendo en la estructura, asociado al mismo concepto y con las mismas características, sin embargo, la función del enlace ha sido adaptada para que no se muestre el contenido del ítem cuando el usuario lo seleccione.

El carácter meramente funcional de las citadas técnicas de adaptación, respaldado por el hecho de que el entorno del sistema es también funcional (el usuario recorre, selecciona e inspecciona los ítems a través de la interfaz de navegación), permite considerar los mecanismos de adaptación al usuario como mecanismos de adaptación por acomodación/aprendizaje enmarcados en el sexto modelo de evolución.

5. CARACTERIZACIÓN ADAPTATIVA DE SEM-HP

En esta sección se caracteriza el modelo SEM-HP desde el punto de vista de la adaptación al usuario, esto es centrándonos en el modelo de evolución 6. Se retoma pues la taxonomía de sistemas hipermedia adaptativos presentada en el capítulo 3.

Para ello, en la tabla 3 se resume el comportamiento de los sistemas hipermedia adaptativos desarrollados de acuerdo a nuestro modelo, desde la perspectiva de los diferentes criterios de clasificación contemplados en dicha taxonomía. Tras lo cuál, se incluye una descripción más detallada de cada punto.

Tabla 3: Caracterización adaptativa de SEM-HP

	MODELO SEM-HP
Aplicabilidad del modelo	General
Objeto de la adaptación	Usuario individual / Grupo de usuarios
Métodos de adaptación	Principalmente adaptación de la navegación
Tipo de prerrequisitos	Prerrequisitos de orden y pedagógicos
Integración de información	Abierto
Interacción usuario-adaptación	Gestión adaptativa y adaptable del modelo de usuario
Creación de hiperdocumentos	Dinámico en la composición de los resúmenes
Información contextual	Ninguna

- **Aplicabilidad del modelo**:

SEM-HP permite la creación de sistemas hipermedia de aplicación *general*. Esto es posible debido a que el dominio de conocimiento no está predefinido, siendo el diseño y la representación del dominio conceptual y de información responsabilidad de cada autor.

En este sentido, el autor tiene total libertad, puede partir desde cero o reutilizar ontologías previas definidas. En cualquier caso, la representación del dominio de conocimiento se lleva a cabo utilizando como herramienta semántica la estructura conceptual.

Además, la diversidad de modos de navegación permite que el modelo sea útil en campos de aplicación muy dispares como por ejemplo: manuales informativos, cursos educativos o simplemente organización no jerárquica de documentos.

- **Objeto de la adaptación**:

La adaptación realizada en los sistemas desarrollados de acuerdo a SEM-HP, depende directamente de las características de cada usuario, representadas internamente en su correspondiente instancia del modelo de usuario.

Esta *adaptación individual* comienza con la elección personalizada de la estructura de navegación y continúa durante su recorrido, de forma sensiblemente distinta dependiendo del modo de navegación que se utilice:

- En la navegación por conceptos (capítulo 16) la adaptación se centra en la generación de resúmenes con una estructura ajustada a los gustos que el usuario especifica durante el estudio de éstos.

- En la navegación por relación conceptual (capítulo 17) la adaptación se encarga de dirigir la navegación del usuario en función de los ítems que ha visitado anteriormente.

- En la navegación por conocimiento (capítulo 18), la adaptación adquiere su mayor expresión, restringiendo y aconsejando al usuario durante su recorrido de acuerdo al estado de conocimiento que posee, sus intereses, meta y preferencias.

Además, durante la navegación libre, el sistema pone en marcha un mecanismo que permite dar soporte semi-automático a una adaptación propiciada por la navegación de un *grupo de usuarios* y no de uno sólo de éstos. Para ello, el sistema estudia el comportamiento colectivo de los usuarios. Y basándose en este análisis sugiere un conjunto de modificaciones (capítulo 21, Retroalimentación adaptativa) que en caso de ser llevadas a término por el autor, harán evolucionar las estructuras de navegación al uso que de éstas se hace mayoritariamente. Se trata, por tanto, de una adaptación que es percibida de modo individual, pero que se realiza a partir del modelo de navegación trazado por un grupo de usuarios.

- **Métodos de adaptación**:

La mayoría de los métodos (consejo local, consejo global, soporte a la orientación, vistas personalizadas,...) y técnicas (ocultación, anotación y deshabilitación de enlaces, rutas guiadas,...) empleadas para adaptarse al usuario se centran en *personalizar la estructura de enlaces*.

Los métodos para adaptar la presentación de la información son menos necesarios debido a que la información se encuentra desgranada a nivel de ítem. Únicamente durante la navegación por conceptos, en la composición de los resúmenes, se utilizan técnicas que permiten al usuario reordenar, expandir y contraer los fragmentos del documento proporcionado, con la posibilidad de personalizar a su gusto la estructura de composición de los resúmenes posteriores.

- **Tipo de prerrequisitos**:

Dependiendo del modo de navegación que ejercite el usuario, los prerrequisitos que utiliza el sistema para determinar la accesibilidad de un ítem son muy distintos. En la navegación libre, tradicional o por conceptos, no existe ningún prerrequisito, de manera que es posible visitar los ítems en cualquier orden.

En la navegación por relación conceptual los prerrequisitos establecidos en las reglas de orden obedecen a las relaciones semánticas existentes entre los conceptos, por lo que dependen del último ítem visitado y del concepto al que éste se asocia (en ocasiones también de si se ha visitado o no anteriormente algún otro ítem requerido). Son, por tanto, un mecanismo de ordenación conceptual, que aunque tiene cierto valor pedagógico no entra de lleno en esta consideración, ya que no tiene en cuenta el conocimiento del usuario.

En la navegación por conocimiento los *prerrequisitos* son indudablemente *pedagógicos*. En este caso, el sistema determina si la visita a un ítem está prohibida, permitida o aconsejada dependiendo del estado de conocimiento que posee el usuario sobre otros ítems que el autor considera relacionados con éste desde un punto de vista didáctico.

Concretamente, las reglas de conocimiento combinan prerrequisitos pedagógicos de accesibilidad e idoneidad (capítulo 8). Los primeros son usados para determinar si el usuario está preparado para asimilar la información contenida en un ítem. Mientras que los segundos permiten establecer si la lectura de un ítem es significativa para el usuario o se trata de información que no le va a aportar nada nuevo.

- **Integración de la información**:

Los sistemas hipermedia desarrollados de acuerdo al modelo SEM-HP son *abiertos*. En cualquier momento el autor puede integrar nuevos ítems en la estructura conceptual de memorización y mostrarlos en todas aquellas presentaciones que desee, con total independencia de su origen.

Para incluir un nuevo ítem, el autor debe proporcionar una *url* o equivalente, que permita localizar inequívocamente el trozo de información deseado. Esta información puede ser de cualquier tipo: texto, audio, video, imagen, ejecutable,... y estar ubicada en el propio PC o en Internet.

Una vez que el autor especifica la dirección del ítem de información se despreocupa del origen del mismo, centrándose en ligarlo adecuadamente a todos aquellos conceptos con los que tiene una relación funcional y especificar en cada caso el tipo de la asociación, esto es, el rol.

- **Interacción del usuario con la adaptación**:

La gestión que se hace en SEM-HP del modelo de usuario es adaptable a la vez que adaptativa. Ciertas características del modelo se actualizan automáticamente a medida que el usuario navega, como por ejemplo su estado de conocimiento (*gestión adaptativa*). Mientras que otras deben ser establecidas explícitamente por el propio usuario, como es el caso del subdominio de conocimiento que le interesa navegar (*gestión adaptable*).

Para la mayoría de las entradas del modelo de usuario que se solicitan explícitamente, el sistema es capaz de realizar una actualización automática, sin embargo se prefiere la explícita. De modo inverso, las entradas que el sistema actualiza automáticamente pueden ser modificadas por el usuario cuando considere que la inferencia es errónea. No obstante, esta capacidad puede bloquearse a petición del autor según la aplicación y el tipo de usuarios del sistema.

Puesto que el usuario tiene en sus manos cierto control sobre la gestión del modelo de usuario, también *controla, en parte, la adaptación* realizada, ya que ésta depende directamente de la información almacenada en el modelo. Además, el usuario puede

configurar parcialmente el tipo de adaptación que desea percibir, eligiendo uno entre los cuatro modos de navegación que se le ofertan.

- **Creación de hiperdocumentos**:

Los ítems de información existen de forma previa a que el usuario los solicite, por lo que en este sentido los sistemas hipermedia desarrollados siguiendo el modelo SEM-HP son estáticos.

Sin embargo, durante el modo de navegación por conceptos se contempla la posibilidad de generar bajo demanda los resúmenes de los conceptos requeridos. Esto es, en el preciso momento en que el usuario selecciona un concepto, el sistema compone dinámicamente un documento con toda la información disponible acerca del mismo.

Esta información se obtiene desde los ítems de información asociados funcionalmente al concepto, y se organiza de acuerdo a lo especificado en una estructura de composición, definida inicialmente por el autor y personalizada a posteriori por el propio usuario.

CAPÍTULO 24

Trabajos Relacionados

Resumen

Se analizan en este capítulo algunos trabajos que guardan cierta similitud con nuestra propuesta, indicando en cada caso las analogías y las carencias que muestran con respecto al modelo SEM-HP. Finalmente, de forma general, se identifican y resumen cuatro puntos clave por los que consideramos que nuestro trabajo es interesante y necesario. Además, para concluir la comparativa, los trabajos estudiados se caracterizan desde un punto de vista adaptativo y evolutivo, como ya se hizo con el modelo SEM-HP en el capítulo anterior. Por otro lado, este capítulo constituye un complemento perfecto del capítulo 3 donde se hace una revisión introductoria de los Sistemas Hipermedia Adaptativos.

Tabla de contenidos

Trabajos Relacionados

1. INTRODUCCIÓN

Obviamente, son muchos los trabajos realizados en el campo de los sistemas hipermedia adaptativos, con lo cual sería casi imposible realizar una comparativa razonable con cada uno de ellos. En su defecto, este capítulo recoge sólo unos cuantos trabajos seleccionados de entre todos los revisados en la literatura científica. Esta selección ha sido realizada atendiendo a dos razones fundamentales: primero que se trate de un trabajo significativo en el área y segundo que sea suficientemente afín con nuestro trabajo como para poder discutir coherentemente las similitudes y diferencias entre ambos.

2. ADAPTS (PETER BRUSILOVSKY, ET AL.)

2.1 Descripción del trabajo

El proyecto ADAPTS (*Adaptive Diagnostics and Personalized Technical Support*) proporciona un sistema de ayuda adaptativo e inteligente para el mantenimiento de un equipo complejo [Brusilovsky, 99] [Brusilovsky, 02]. Para ello, crea y mantiene un modelo de usuario que almacena el conocimiento, experiencia y preferencias del técnico.

Basándose en dicho modelo y en la actividad actual del técnico, monotorizada por las respuestas que éste da al sistema, se determina una estrategia de diagnóstico, esto es, un conjunto de tareas que el técnico debe realizar. Además, el sistema proporciona información técnica contextual relevante para cada tarea, y la ayuda de navegación necesaria para llevarlas a cabo.

A nivel de arquitectura y proceso de trabajo se pueden distinguir dos subprocesos:

a) Interacción adaptativa con el técnico.

b) Generación del diagnóstico adaptativo.

Este último subproceso es realizado por el *motor de diagnóstico* a partir del modelo del sistema, las herramientas disponibles, la información del modelo de usuario, la historia de mantenimiento del equipo y la configuración actual del mismo. El resultado es una secuencia de tareas que el técnico debe realizar en el orden dado, para cada tarea se proporciona información técnica comprensible por el técnico.

La interfaz hipermedia es adaptativa y presenta dos marcos: de perfil y de presentación de contenido (véase la figura 1).

1. **Marco de perfil**: Muestra una lista de tareas. Esta lista puede ser navegada por el técnico, de modo que cuanto mayor sea su experiencia más libertad se le concede para moverse por la lista. En el caso de técnicos novatos no se permite la navegación para evitar pérdidas y confusiones. Además, en este marco se indica el estado de

realización de cada tarea y una lista de comprobaciones o subtareas, que para los técnicos expertos aparece contraída.

2. **Marco de presentación de contenido**: Visualiza información relevante para la tarea seleccionada en el marco de perfil. El contenido de este marco es adaptado en varios sentidos:

   - Expandir y contraer párrafos procedurales para revelar u ocultar detalles según el grado de formación del técnico.
   - Seleccionar los enlaces que se ofrecen al técnico en función de la experiencia de éste.
   - Indicar la relevancia de los enlaces ofrecidos de forma visual (iconos o colores diferentes), textual (anotaciones o comentarios) o mediante la ordenación de dichos enlaces.

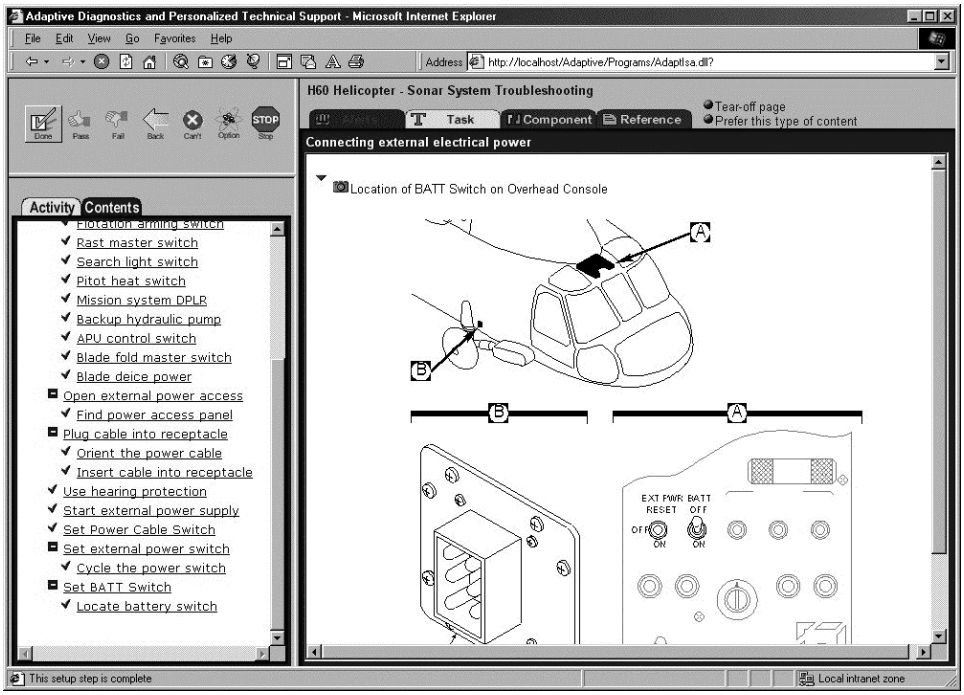


Figura 1: Interfaz ADAPTS

La representación del conocimiento se realiza mediante un *modelo de dominio* que integra dos jerarquías de conceptos: una de componentes y otra de tareas. En el *modelo de usuario* se almacena información sobre la experiencia y conocimiento que el técnico tiene sobre cada concepto (tarea o componente) del modelo de dominio.

Además, se asocia un peso a cada aspecto del modelo de usuario y de esta forma se calcula, mediante un polinomio de pesos, la capacidad que tiene el técnico para realizar una tarea.

La conexión entre los documentos y el modelo de dominio se realiza mediante un proceso de *indexación*. Existen dos tipos de indexación de contenidos:

a) Indexación basada en la función: Un documento se conecta mediante enlaces tipificados con todas las componentes involucradas en él. El tipo del enlace indica el tipo de función.

b) Indexación de tareas: Se indexa un documento con una tarea si éste explica cómo realizarla.

Por supuesto, el contenido de los documentos es multimedia: fotos a color, videos de entrenamiento, animaciones, simulaciones, etc.

2.2 Relación con nuestro trabajo

La principal diferencia con nuestro modelo, es que el sistema ADAPTS tiene un campo de aplicación específico y restringido mientras que SEM-HP es un modelo general que permite el desarrollo de sistemas hipermedia muy diferentes.

En ADAPTS los conceptos del modelo de dominio versan en torno a un tema muy concreto, sólo se admiten dos tipos de conceptos: componentes y tareas técnicas. Por el contrario en SEM-HP la noción de concepto da cabida a cualquier idea etiquetada, independientemente del área de conocimiento donde se integre.

Otra diferencia importante es que el modelo de dominio de ADAPTS sólo permite relaciones jerárquicas entre conceptos del mismo tipo (árboles de componentes y árboles de tareas). En cambio, en SEM-HP los conceptos se relacionan entre sí, sin distinción de tipo y en red. Además, en esa misma red se integran los ítems de información sin necesidad de utilizar un mecanismo adicional de indexación como en el caso de ADAPTS.

En el modelo de usuario de ADAPTS prima el conocimiento y la experiencia del técnico en las tareas y componentes representadas, mientras que en SEM-HP, el modelo de usuario (capítulo 12) mantiene, además, otra información (preferencias, gustos, intereses y metas), que permite una adaptación más amplia y general. Asimismo, en SEM-HP no sólo se calcula el conocimiento del usuario acerca de los conceptos, también de los ítems.

A pesar de las citadas diferencias podemos encontrar varios puntos en común con nuestro trabajo. Por ejemplo, la generación del diagnóstico adaptativo realizado en ADAPTS coincide con las rutas guiadas propuestas en SEM-HP (capítulo 20), en el sentido de que el sistema construye una secuencia (de tareas en ADAPTS y de visitas a ítems en SEM-HP) que tras ser seguida en el orden especificado permite al usuario (técnico en ADAPTS) alcanzar su objetivo (arreglar el equipo en ADAPTS o conseguir un determinado estado de conocimiento en SEM-HP).

En ambos sistemas se restringe la navegación del usuario en función de su conocimiento: en ADAPTS se trata de limitar el orden en que el técnico consulta las tareas y en SEM-HP de impedir la lectura de determinados ítems considerados inaccesibles por las reglas de orden o de conocimiento (capítulo 8). Del mismo modo, en ambos sistemas se realiza una anotación especial de ciertos enlaces para atraer la atención del usuario hacia ellos (ítems deseables en SEM-HP (capítulo 19)).

Además, en ambos se utiliza una técnica de adaptación que permite expandir u ocultar fragmentos en un documento. En ADAPTS esta técnica se aplica en todos los documentos para evitar que el técnico se pierda con información demasiado compleja. En SEM-HP esto se resuelve a nivel de ítem, impidiendo durante la navegación por conocimiento el acceso a aquellos ítems para cuya comprensión el usuario no está preparado. De modo que la citada técnica se utiliza principalmente por cuestiones de espacio en la generación de los resúmenes de conceptos (capítulo 16).

3. EL PROYECTO PUSH (KRISTINA HÖÖK, ET AL.)

3.1 Descripción del trabajo

El proyecto PUSH (*Plan- and User Sensitive Help*) implementa un sistema hipertexto, multi-modal, altamente interactivo y adaptativo, que proporciona información sobre métodos de desarrollo del software [Espinoza, 96] [Höök, 98]. Los métodos están formados por: procesos y objetos. Los procesos son actividades que tienen lugar durante las fases de desarrollo de un proyecto. Y los objetos son los resultados de los métodos, esto es, la entrada y salida de los procesos.

Los objetos se relacionan entre sí mediante relaciones del tipo es_un o parte_de. Los procesos también se relacionan unos con otros para indicar que: un proceso es parte de otro, un proceso debe ejecutarse antes de otro o la ejecución de dos procesos debe ser paralela.

Cada página que el sistema presenta al usuario tiene tres marcos claramente diferenciados (véase la figura 2):

1. **Descripción textual** del método.

2. **Grafo** que muestra el proceso actual, sus super-procesos, sub-procesos y objetos de entrada y salida.

3. **Guía de la descripción textual** que contiene las cabeceras de las unidades de información que constituyen la descripción (se marcan en negrita las unidades disponibles).

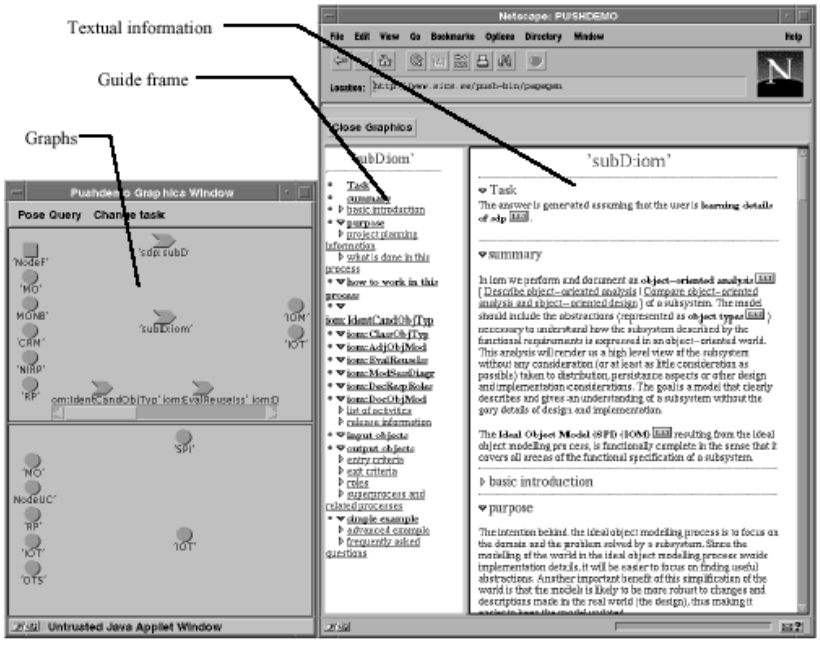


Figura 2: Interfaz PUSH

Los grafos no son fotografías estáticas almacenadas en disco, sino que se generan a partir de una base de datos que contiene toda la información y cualquier cambio en ésta

es inmediatamente reflejado en el grafo. El grafo proporciona una vista comprensible del espacio de información a partir de la posición actual y, además, permite al usuario navegar haciendo *clic* sobre los objetos del grafo.

Cada objeto del grafo (proceso u objeto de desarrollo del software) tiene una serie de atributos, entre los que se encuentran las entidades de información. Una entidad de información no es otra cosa que un trozo de información que describe un determinado aspecto del objeto. Para cada objeto existe un único nodo que contiene toda la información sobre éste, esto es, todas las entidades de información. Este nodo se genera dinámicamente, de modo que sólo la página de inicio se almacena en disco.

Normalmente no se visualizan simultáneamente todas las entidades de un nodo. El usuario puede cerrar o abrir entidades de información haciendo *clic* sobre ellas en el marco guía de la descripción textual. Cuando se hace *clic* sobre una entidad no visualizada, su texto se inserta en el marco de descripción textual (un nuevo *clic* la oculta).

Los conceptos cruciales para la comprensión del método (*hotwords*), son resaltados en negrita en la descripción textual del objeto. Junto a cada *hotword* aparece un botón que al pulsarse despliega una lista de preguntas a cerca del concepto. La respuesta a la pregunta seleccionada se inserta en el marco de texto.

Además, el usuario puede navegar componiendo preguntas a través de menús, que serán respondidas por el sistema con la generación de una nueva página. Es una alternativa a la navegación por grafos que permite conocer las necesidades de los usuarios más avanzados.

En el enfoque expuesto no se contempla adaptación explícita al conocimiento ni a las preferencias del usuario. Esto se debe a que:

a) Al permitir la manipulación de la página de respuesta, en concreto el uso de los ***hotwords***, se abastece la diferencia de conocimiento de los usuarios. Cada usuario hará uso de unas preguntas u otras en función de su conocimiento.

b) La diferencia de unos usuarios con otros en cuanto a preferencias y características cognitivas se supera con el carácter **multi-modal** del sistema, que presenta dos formas de navegación (grafo y menús) y dos tipos de contenidos (texto y grafo).

3.2 Relación con nuestro trabajo

De nuevo, la comparación con nuestro trabajo requiere cierto proceso de abstracción, debido a que el sistema PUSH resuelve un problema muy concreto, mientras que SEM-HP es un modelo de aplicación general.

Aunque en ambos trabajos se permite la navegación sobre estructuras no jerárquicas, grafos en PUSH y redes semánticas en SEM-HP, las diferencias que existen entre ellos son notables.

En PUSH los únicos objetos que se pueden contemplar en el grafo son procesos y objetos SDP (*Software Development Process*). Esto hace necesario resaltar dentro de la descripción textual de un objeto los conceptos importantes que aparecen (*hotwords*). En

SEM-HP, la red semántica admite conceptos de muy diversa índole, esto es, se puede representar cualquier concepto como un nodo en la red. De este modo, primero se identifican los conceptos importantes, y luego, se asocian a éstos los ítems de información que los describen, no al revés. Además, la relación entre la unidad de información y el concepto se representa gráficamente sobre la propia red a través de las asociaciones funcionales.

Por otro lado, las relaciones conceptuales que se utilizan en PUSH son mucho más restrictivas que en SEM-HP. Esto es, las relaciones posibles entre objetos (es_un, parte_de), entre procesos (super_proceso, sub_proceso,...) y entre ambos (entrada, salida), están predefinidas y limitadas en PUSH. Mientras que en SEM-HP el autor puede usar las relaciones predefinidas (parte_de, es_un,...) o especificar semánticamente aquellas que necesite.

Tampoco quedan muy claras en PUSH las relaciones existentes entre los métodos, ya que el grafo es una representación parcial centrada en un método concreto y no parece haber ningún elemento integrador. En SEM-HP este problema podría resolverse creando a partir de la estructura conceptual completa una presentación para cada método de desarrollo. De esta forma, se mantiene en el Sistema de Memorización una representación que relaciona los distintos métodos, y que además muestra la vinculación de un mismo proceso u objeto con varios métodos distintos.

Podemos decir que los grafos utilizados en PUSH representan básicamente flujos de trabajo donde se indica el orden de ejecución de los procesos y las entradas y salidas de los mismos, frente a la red semántica de SEM-HP que representa conocimiento y obedece, por tanto, a una noción diferente y más amplia de relación conceptual.

El sistema PUSH apuesta por una navegación multi-modal, permitiendo navegar al usuario sobre el grafo y a través de menús. En SEM-HP la navegación multi-modal ofrece cuatro modos, en los que varía la estructura de navegación pero, también, las restricciones y los métodos de adaptación aplicados (capítulo 15, Modos de navegación).

La navegación realizada en PUSH puede equipararse a la navegación por conceptos propuesta en SEM-HP. Ya que el usuario selecciona un objeto del grafo (concepto en SEM-HP) y obtiene un documento compuesto con las entidades de información asociadas (ítems en SEM-HP). También se aplica la técnica de adaptación que permite expandir o contraer entidades en el documento.

Respecto a la adaptación realizada, ambos son sistemas interactivos que permiten al usuario modificar las asunciones realizadas automáticamente por el sistema. Sin embargo, la adaptación realizada en PUSH es más limitada que la que se lleva a cabo en SEM-HP, donde sí se tiene en cuenta el conocimiento y las preferencias del usuario.

En PUSH la adaptación al conocimiento del usuario consiste en proporcionarle una lista de preguntas sobre cada concepto importante y que él elija aquellas que le interesen. El problema es ¿cómo sabe el sistema que el usuario se encuentra en disposición de comprender la respuesta a una pregunta si desconoce el conocimiento que éste posee? Si para entender la respuesta, de nuevo debe solicitar otras preguntas, es muy probable que un usuario inexperto padezca sobrecarga cognitiva.

En SEM-HP, el sistema es capaz de controlar la adquisición de conocimiento del usuario mientras navega, y dirigir su recorrido impidiendo que se abrume leyendo información que no es capaz de asimilar y que pierda el tiempo con información redundante para él.

En PUSH la única preferencia que el usuario puede especificar es el modo de navegación, grafo o menús. En SEM-HP además del modo de navegación, el usuario puede establecer sus gustos acerca de los ítems de información y el subdominio de conocimiento que más le interesa. Usando esta información el sistema elige la estructura de navegación más apropiada, genera rutas guiadas personalizadas y resalta los ítems cuyas características se ajustan mejor a los gustos del usuario.

4. AHAM Y AHA (PAUL DE BRA, ET AL.)

4.1 Descripción del trabajo

4.1.1 El modelo AHAM

AHAM es un modelo para el desarrollo de aplicaciones hipermedia adaptativas, que ha sido diseñado específicamente para desarrollar fácilmente herramientas autoras de hipermedia adaptativo [DeBra, 98] [Wu, 00].

El modelo AHAM (*Adaptive Hipermedia Application Model*) es una extensión del modelo de referencia Dexter [Halasz, 90]. En AHAM la capa de almacenamiento definida como parte fundamental del modelo Dexter está formada por el modelo de dominio, el modelo de usuario y el modelo de adaptación.

Concretamente, en AHAM una aplicación hipermedia se divide en cuatro partes: modelo de dominio (MD), modelo de usuario (MU), modelo de adaptación (MA) y motor de adaptación.

- **Modelo de dominio**:

Este modelo describe como se estructura el *dominio de aplicación*. Es la visión que el autor tiene del dominio de aplicación expresada en términos de conceptos. Los principales elementos que aparecen son:

- Anclaje: Se define mediante una pareja <identificador, punto_final_del_anclaje>. El punto final del anclaje es una localización dentro de un concepto compuesto.
- Componente: Es una tupla <identificador, información>. La información del componente es un conjunto de atributos, una lista de anclajes y una especificación de presentación.
- Concepto: Es una componente que representa un ítem abstracto del dominio de aplicación. Hay conceptos atómicos (fragmentos de información que no admiten adaptación) y conceptos compuestos (una secuencia de conceptos hijos más una función constructor). Un concepto compuesto donde todos sus hijos son fragmentos se llama página. La jerarquía de conceptos debe ser un grafo dirigido acíclico y cada concepto atómico debe pertenecer a un concepto compuesto.

- Relación entre conceptos: Es una tupla <c1, c2, T, A> donde c1 y c2 son dos conceptos, T es el tipo de la relación y A es el valor de un atributo que depende del tipo.
- Tipo de relación: c1 parte_de c2, c1 prerrequisito_para c2, c1 inhibe_a c2 y c1 enlace_a c2. Cada tipo de relación tiene un atributo asociado. El valor de éste indica: para el primer tipo (parte_de) la fracción de c2 que es representada por c1, para el segundo tipo (prerrequisito_para) el conocimiento mínimo que se debe tener sobre c1 para que leer c2 sea deseable, para el tercer tipo (inhibe) el conocimiento máximo que se puede tener de c1 para que siga siendo deseable leer c2, y para el cuarto tipo (enlace_a) si el enlace es externo, estático o adaptable.

- **Modelo de usuario**:

Describe el *conocimiento* que tiene el usuario sobre los conceptos del dominio de aplicación. Relaciona por tanto al usuario con el modelo de dominio. Para cada usuario se mantiene una tabla que almacena para cada concepto el valor de sus atributos. A la estructura de la tabla se le llama esquema del modelo de usuario y al contenido de la tabla para un usuario concreto se le conoce como instancia del modelo de usuario.

En el modelo de usuario no sólo se almacenan los conceptos del dominio de aplicación sino otras características del usuario como preferencias, metas, experiencia en la materia y experiencia en la navegación de hipermedia. Para los conceptos del dominio de aplicación se recogen los siguientes datos: el nivel de conocimiento del usuario, relevancia, si ha leído algo sobre el concepto, si está preparado para entenderlo, etc.

- **Modelo de adaptación**:

En las primeras versiones de AHAM se llamó modelo “de enseñanza” por su principal aplicación en el área educativa. Determina cómo el conocimiento del usuario influye en la presentación de la información. Esta compuesto por reglas de adaptación (denominadas anteriormente reglas pedagógicas) que pueden dividirse en dos grupos:

- *Reglas de adaptación genéricas* predefinidas por el sistema, que usan variables para representar conceptos y relaciones entre conceptos.

- *Reglas de adaptación específicas* establecidas por el autor, que se utilizan como excepciones y siempre tienen precedencia sobre las genéricas.

Cada regla tiene asociada una fase de ejecución, *pre* si se ejecuta antes y durante la generación de la página o *post* si se ejecuta después, y un atributo *booleano* que indica si la aplicación de la regla puede desencadenar que otras reglas se disparen. Las reglas se dividen en cuatro grupos dependiendo del paso de adaptación al que pertenecen:

a) IU. Inicializan el modelo de usuario.

b) UU-pre. Actualizan el modelo de usuario antes de generar la página.

c) UU-post. Actualizan el modelo de usuario después de generar la página.

d) GA. Generan la adaptación.

En cada grupo se disparan reglas hasta que no haya más reglas aplicables o hasta que su aplicación no produzca cambios.

- **Motor de adaptación**:

Es el entorno software que utiliza las reglas para generar la especificación de la presentación. Sus funciones pueden resumirse en:

- Proporcionar constructores y selectores de páginas, que para cada concepto compuesto determinan que página se visualiza cuando se sigue un enlace hacia él, y para cada página construyen una presentación adaptativa a partir de sus fragmentos.
- Opcionalmente puede presentar un lenguaje sencillo para que el autor pueda describir nuevos constructores.
- Ejecutar los constructores y como consecuencia *realizar la adaptación*.
- Actualizar el modelo de usuario cada vez que éste visita una página (disparar reglas UU-post).

4.1.2 La arquitectura AHA

Las siglas AHA son el acrónimo de Arquitectura Hipermedia Adaptativa (*Adaptive Hipermedia Architecture*). Se trata de un sistema hipermedia adaptativo desarrollado en la Universidad de Tecnología de Eindhoven [DeBra, 98b] [DeBra, 00] [DeBra, 02]. Es un sistema basado en web que usa el lenguaje HTML. El núcleo del sistema AHA (figura 3) consiste en un motor adaptativo que mantiene un modelo de usuario basado en el conocimiento de éste sobre los conceptos del modelo de dominio. Inicialmente surgió para ofrecer un curso adaptativo y un kiosco "virtual", pero ha llegado a convertirse en un sistema hipermedia adaptativo genérico.

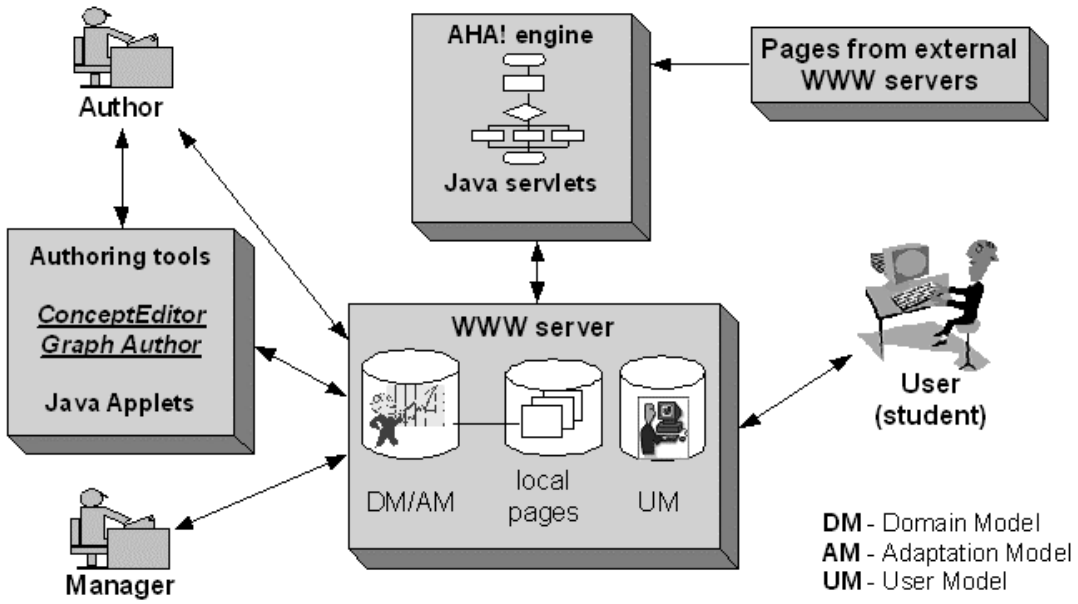


Figura 3: Arquitectura del sistema AHA

- **Modelo de usuario**:

En el modelo de usuario se almacena el conocimiento, metas, preferencias, experiencia previa en la materia y experiencia en el hiperespacio del usuario. Todo esto se representa como un conjunto de variables y una serie de atributos asociados a cada una. Estas variables se denominan de manera genérica conceptos, y el nivel de conocimiento

que tiene el usuario sobre cada concepto surge como un atributo inmediato. Se plantean tres tipos de representaciones para el grado de conocimiento:

- Modelo *booleano*. Se conoce o no se conoce el concepto.
- Modelo discreto. Número pequeño de valores de conocimiento.
- Modelo continuo. Rango de valores.

En las primeras versiones de AHA se utilizaba el modelo *booleano* para representar el grado de conocimiento, sin embargo, éste ha sido sustituido en versiones posteriores por el modelo continuo. Concretamente el conocimiento es representado con un valor porcentual.

- **Reglas de actualización**:

A cada concepto se le asocia una sola regla de actualización. Se distinguen cuatro tipos de actualizaciones en AHA:

- Actualización *monótona*: Sólo es aplicable a conceptos abstractos. La actualización del concepto A implica una actualización relativa y positiva sobre un concepto B, lo que significa que el valor de B se incrementa en un determinado porcentaje del valor de A. El algoritmo se aplica recursivamente sobre B.
- Actualización *no monótona*: Se aplica sólo a conceptos abstractos. La actualización del valor de un concepto implica la actualización relativa y negativa de otro concepto. No se continúa recursivamente con dicho concepto.
- Actualización *absoluta*: Se aplica a conceptos página. La actualización derivada es un valor fijo que se incrementa o decrementa del valor del concepto. El algoritmo no se aplica recursivamente.
- Actualización *propia* (a sí mismo): La actualización del valor de un concepto induce a la actualización del propio concepto. Sólo es aplicable a conceptos página. No es recursiva.

Las reglas de actualización se disparan cuando el usuario solicita acceder a una página y se ejecutan antes de generarla. Por lo tanto, el conocimiento obtenido con la lectura de la página se tiene en cuenta durante la generación de la misma. Esto permite que en la página aparezcan ya anotados los enlaces a otras páginas que se convierten en relevantes después de la lectura de ésta. Sin embargo, hace que no sea posible generar una primera versión de la página, es decir asumiendo el conocimiento del usuario antes de su visita.

Cuando el usuario estudia una página, el conocimiento de ésta influye en un determinado grado en el conocimiento de la sección a la que pertenece, y por lo tanto, en otro rango (lógicamente menor) en el conocimiento del capítulo al que pertenece la sección, y como no, en el conocimiento del libro al que pertenece el capítulo. Para expresar esta cascada de conocimiento se definen las reglas recursivas. Sin embargo, existen varios problemas derivados de esta recursión:

- Pueden aparecer bucles.
- El acceso a una página puede generar dos actualizaciones sobre un mismo concepto.
- Dificultad para decidir cómo se manejan las visitas repetidas a una página.

Para asegurar que la aplicación del algoritmo de actualización termina se proponen dos soluciones:

a) Impedir bucles infinitos y recursiones profundas en las reglas.

b) El motor de adaptación se encarga de asegurar que la transición termina incluso cuando las definiciones de las reglas causen bucles.

La segunda solución es obviamente mejor, pero más difícil de conseguir.

- **Arquitectura de AHA**:

AHA está implementado casi totalmente en Java. Usa HTML como lenguaje para escribir tanto el contenido como los constructores de las páginas. Los fragmentos son trozos de fuente *html* delimitados por comentarios que indican el comienzo y el final del mismo. Además, los comentarios *html* son utilizados y abusados con los siguientes propósitos:

- Cada página está asociada a un concepto que normalmente tiene el mismo nombre que la página. La asociación del concepto y la página se define con un comentario *<!--generates c >* en la cabecera de la página.
- Cada página comienza con un comentario *<!--requires (c1 and c2 and ... cn) >* que establece qué conceptos deben ser conocidos antes de que un enlace a la página se considere deseable.
- Los comentarios *html* son también utilizados por el autor para establecer la condición que debe cumplir un determinado fragmento para visualizarse en la presentación actual de la página.

La cabecera *html* de cada página generada en AHA contiene una definición del estilo de hoja, que establece los colores que deben mostrar los distintos tipos de enlaces (esquema de coloreo de enlaces). Desde el punto de vista de implementación, en AHA se distinguen tres tipos de enlaces, para los que, a continuación, se indica su color por defecto:

- Absolutos o externos. Rojo para los no visitados y gris para los visitados.
- Relativo incondicional. Azul para los no visitados y morado para los visitados.
- Relativo condicional. Se evalúa la expresión "*requires*" asociada a la página destino del enlace y si es *true* se sigue el mismo esquema que para los enlaces relativos incondicionales, en caso de ser *false* el color asociado es negro.

4.2 Relación con nuestro trabajo

Puesto que tanto AHAM como SEM-HP son modelos de aplicación general que pretenden servir para el desarrollo de sistemas hipermedia adaptativos cualquiera que sea su tema y utilidad, cabe esperar un mayor grado de similitud que en los casos anteriores.

En ambos se distingue un modelo de dominio, un modelo de usuario y un modelo de adaptación. Concretamente, en SEM-HP, el modelo de dominio es gestionado entre el Sistema de Memorización y de Presentación, y el modelo de usuario, así como el

modelo de adaptación son responsabilidad del Sistema de Aprendizaje. Las características contempladas en el modelo de usuario son similares, en uno y otro, ya que ambos modelos intentan proporcionar la máxima funcionalidad adaptativa.

En ambos casos se realiza una representación explícita del conocimiento, sin embargo, el tratamiento que se hace del dominio conceptual y de información es muy diferente en cada uno de ellos. En AHAM no existe una distinción clara entre el concepto y la información que lo describe. En SEM-HP, por el contrario, se separa expresamente lo que es la idea etiquetada semánticamente (concepto) de lo que es la información asociada a ésta (ítems).

De este modo, podríamos equiparar el concepto atómico de AHAM con el ítem de información de SEM-HP. Y, en consecuencia, la página de AHAM (composición de conceptos atómicos) con el documento generado como resumen de un concepto en SEM-HP.

El modelo de dominio en AHAM no acepta relaciones cíclicas entre los conceptos, mientras que en SEM-HP sí se permiten puesto que este tipo de relaciones son muy habituales en la vida cotidiana. Además, mientras que en SEM-HP no existen restricciones respecto al tipo de la relación conceptual, en AHAM existen cuatro tipos de relaciones predefinidas: “composición”, “prerrequisito”, “inhibición” y “enlace”, no pudiéndose establecer ninguna fuera de éstas.

Otra diferencia significativa es que en AHAM, a través de las relaciones de tipo “prerrequisito” e “inhibe”, el autor integra en el modelo de dominio las restricciones de navegación existentes entre los conceptos. Esto es, mezcla la representación de la información y el orden (parcial) en que ésta debe ser recorrida. En SEM-HP los requisitos pedagógicos necesarios para visitar un ítem de información se especifican fuera del modelo de dominio, concretamente en las reglas de conocimiento del Sistema de Aprendizaje. Esta separación permite mayor flexibilidad a dos niveles:

1. Para un mismo ítem se pueden tener varias reglas de conocimiento, de modo que basta con que el usuario siga el camino trazado en una de ellas.

2. Sobre un mismo modelo de dominio es posible superponer varios esquemas de navegación, esto es distintos conjuntos de reglas, eligiéndose uno u otro en función del perfil del usuario.

En ambos modelos, el autor puede especificar el conocimiento mínimo para que la lectura de una página esté permitida y el conocimiento máximo para que dicha lectura sea deseable. En AHAM el autor tiene que separar explícitamente las dos clases de requisitos pedagógicos en relaciones del tipo “inhibe” o “prerrequisito”. Por el contrario, en SEM-HP el autor escribe las condiciones necesarias para la visita de la página, sin preocuparse de la diferencia entre una condición de tipo “inhibe” y otra de tipo “prerrequisito”. Después, el sistema se encarga de separar, automáticamente, las restricciones impuestas por el autor en restricciones de accesibilidad (prerrequisito) y restricciones de idoneidad (inhibe).

Otro aspecto que separa bastante los dos modelos, es la forma de recorrer la información ofrecida. En SEM-HP la navegación se realiza sobre la propia red semántica con el soporte de orientación que esto proporciona. En AHAM, el modelo de

dominio se utiliza para generar las páginas pero no permite navegar directamente sobre él, de modo que, finalmente, la navegación se realiza a través de los enlaces insertados en la página actual. Además, AHAM a diferencia de SEM-HP no admite modos alternativos para navegar.

La representación del conocimiento es también distinta. En AHA se utiliza un valor porcentual, esto es, un número entero de 0 a 100 para indicar cuánto conoce el usuario acerca de un concepto. En SEM-HP se establecen cinco etiquetas semánticas que permiten al autor definir las reglas de aprendizaje de una forma suficientemente precisa, y más intuitiva que un porcentaje. Este menor número de grados de conocimiento permite reducir considerablemente el espacio de estados posibles (muy importante para la generación de rutas guiadas): 5^n estados en SEM-HP frente a 100^n en AHAM.

Por último, una diferencia importante entre ambos modelos es el modo en que se actualiza el conocimiento del usuario. Ambos incorporan un mecanismo denominado reglas de actualización (capítulo 9), sin embargo, éste es muy distinto en uno y otro. Las principales diferencias son las siguientes:

- AHAM distingue actualización de conceptos atómicos (ítems en SEM-HP) y actualización de conceptos abstractos. En SEM-HP las reglas de actualización sólo se definen para los ítems. El conocimiento de un concepto se calcula, mediante un mecanismo aparte, las reglas de peso, que combina el conocimiento de los ítems asociados a éste.

- AHAM concibe un tipo de actualización negativa. Por el contrario, en SEM-HP toda actualización es no negativa, ya que tras leer un ítem no tiene sentido que el conocimiento del usuario disminuya (ni sobre el ítem leído ni sobre otro cualquiera), en el peor caso que se mantenga.

- En AHAM las reglas de actualización no distinguen si se trata de la primera visita al concepto o es una visita repetida. En SEM-HP el autor puede especificar que una actualización sólo se ejecute si es la primera vez que el usuario visita el ítem.

- En AHAM la ejecución de las reglas de actualización es recursiva, lo cual plantea serios problemas: bucles, dobles actualizaciones, etc. En SEM-HP la regla de actualización de un ítem se ejecuta incrementando el conocimiento del usuario acerca de todos los ítems que corresponda, sin ejecutar recursivamente la regla de actualización asociada a cada uno de ellos. Esto es posible, gracias a la clara distinción entre ítem y concepto que permite disponer de dos mecanismos de actualización independientes: las reglas de actualización y las reglas de peso.

5. ENFOQUE DE JOHN BOLLEN (JOHN BOLLEN, ET AL.)

5.1 Descripción del trabajo

El enfoque de este trabajo [Bollen, 98] [Bollen, 00] se centra en utilizar modelos de grupos de usuarios para hacer recomendaciones de hiperenlaces personalizadas a cada uno de ellos. Para esto se apoya en las siguientes consideraciones:

a) A los usuarios que buscan información dentro de un dominio que le es desconocido le resulta más beneficioso el conocimiento de otros usuarios familiarizados con éste que una interfaz personalizada. Por lo tanto, es sensato utilizar el conocimiento de un grupo de usuarios para aconsejar a un usuario particular.

b) Las necesidades de información de los usuarios y sus características personales son factores altamente inestables y difíciles de medir. Para evitar este problema, no se miden las características individuales de los usuarios, sino que se buscan los patrones de navegación del grupo.

c) La gente, por lo general, tiene ideas predefinidas y estables sobre las asociaciones entre conceptos y esas ideas se superponen entre grupos de individuos. Los diseñadores usan sus nociones de asociación para enlazar las páginas hipertexto y los usuarios para determinar como navegarlas. Si los modelos de los usuarios y de los diseñadores se superponen de forma pobre, la interacción del usuario con el sistema va a ser ineficiente, luego es bueno dejar que los usuarios seleccionen las relaciones entre conceptos implícitamente mediante el uso del navegador.

El enfoque contempla estos tres puntos de la siguiente forma: *El propio sistema reestructura su red hipertexto mientras que está siendo navegada*. Para ello asocia un peso de conexión a los hiperenlaces y ejecuta tres reglas de aprendizaje que modifican convenientemente esos pesos. Los cambios en la red se muestran a los usuarios a través de la ordenación de los enlaces presentados. Concretamente, los enlaces que salen de un nodo se ordenan de forma descendiente según su peso.

El peso de conexión sirve para modular la relevancia del enlace, de manera que cuando más alto es el peso de un enlace más fuerte es la asociación entre los conceptos que une. Se usan tres **reglas de aprendizaje** que dinámicamente cambian la estructura de enlaces de la red de acuerdo a los caminos seguidos por los usuarios. A través de estas reglas, un grupo de usuarios mediante sus patrones de actividad superpuestos reforzará conexiones específicas y gradualmente reestructurará la estructura de la red (véase la figura 4).

- **Regla de aprendizaje de frecuencia:**

Se basa en la ley que modela el aprendizaje humano y establece que “si dos conceptos son simultáneamente activados la conexión entre ellos se refuerza”. Cada regla de aprendizaje tiene una recompensa asociada, que en este caso recibe el nombre de recompensa de frecuencia (Rf). De esta forma si se sigue un hiperenlace entre un nodo A y un nodo B, el peso del enlace A-B se incrementa en Rf.

- **Regla de aprendizaje de transitividad**:

Cuando un usuario navega de un nodo A hacia un nodo B y a continuación a un nodo C se refuerza la conexión entre A y C. El incremento correspondiente viene especificado por la recompensa de transitividad (Rt) que debe ser menor que la recompensa de frecuencia. Esta regla tiene como objetivo acortar los caminos de navegación. La conexión A-C puede ser nuevamente reforzada, en un futuro, a través de la regla de frecuencia.

- **Regla de aprendizaje de simetría**:

Se refuerza la conexión B-A si la conexión entre A-B ha sido atravesada. La recompensa asociada recibe el nombre de recompensa de simetría (Rs) y será menor que la recompensa de transitividad.

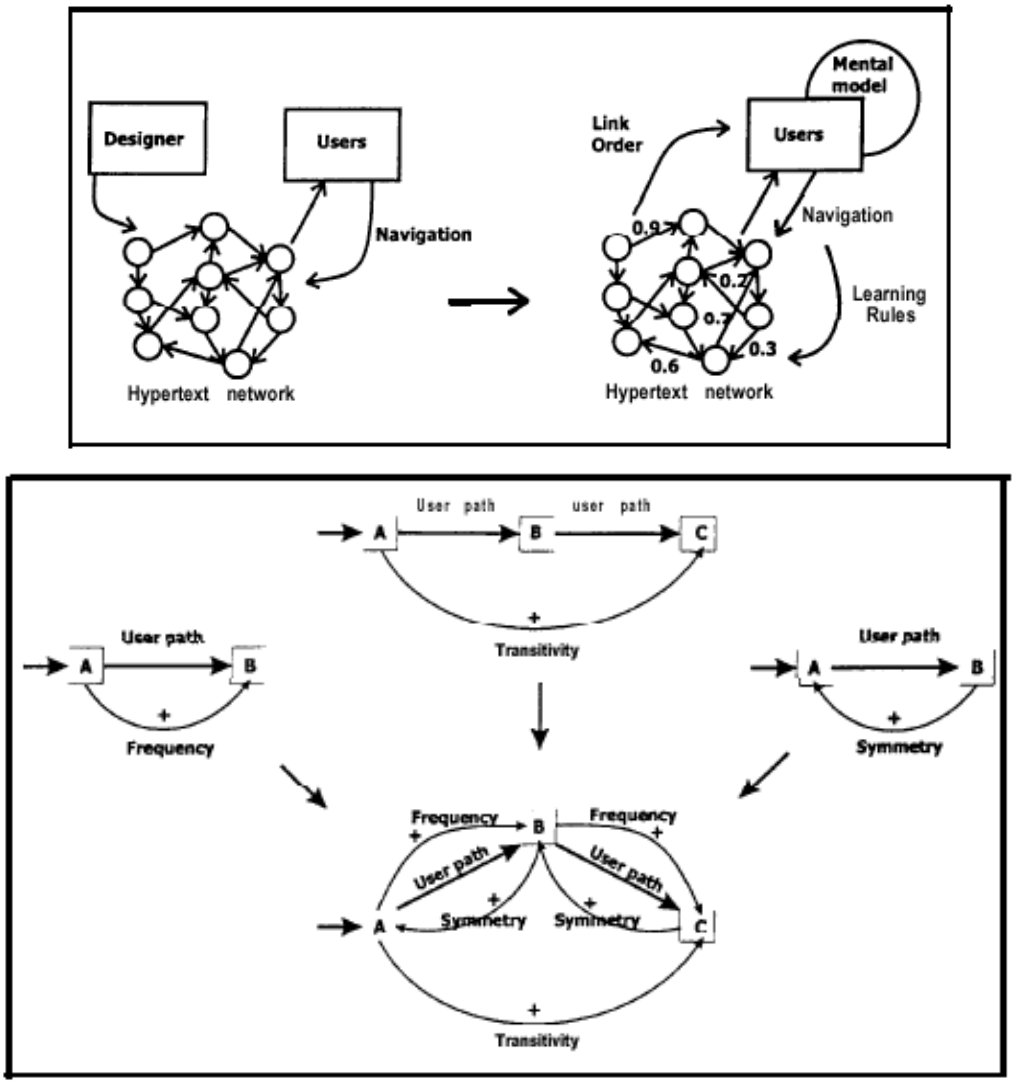


Figura 4: Enfoque de Johan Bollen

El sistema puede ser consultado de forma tradicional, esto es con una consulta asociativa (el usuario proporciona palabras clave y el sistema devuelve los nodos donde éstas encajan) o por *activación directa*. En el último caso, el usuario proporciona un conjunto de nodos que asociativamente expresan el significado de los ítems que quiere recuperar, es decir, los caminos de recuperación del usuario pueden usarse como consultas. La activación extensiva se compone de dos fases:

1. Inicio. Se activa un conjunto de nodos (consulta).
2. Extensión. La energía de la activación se extiende desde esos nodos a todos los nodos relacionados con ellos, la extensión está modulada por los pesos de los enlaces. La formula sería $At+1 = At * M / a_i = \sum a_j * w_{ij}$, donde A es el vector de activación, M la matriz de pesos, a_i es la activación del nodo i y w_{ij} es el peso del enlace entre el nodo i y el nodo j.

3. Resultado. El sistema devuelve una lista ordenada de los nodos que se han activado con los valores más altos.

5.2 Relación con nuestro trabajo

A pesar de que a priori este trabajo parece no tener demasiado en común con el nuestro, ha sido incluido en la comparativa porque si nos centramos en el mecanismo de retroalimentación adaptativa implementado en SEM-HP (capítulo 21) encontramos cierta relación con el enfoque descrito.

Efectivamente, la idea última es la misma en ambos: analizar el comportamiento del grupo de usuarios para reestructurar la red hipermedia, con la intención de que ésta contemple las relaciones definidas por el grupo durante su navegación. No obstante, las diferencias que existen en la forma utilizada para aplicarla son considerables. Primeramente, en SEM-HP, la navegación del grupo de usuarios no se analiza siempre, únicamente cuando se trata de un modo de navegación libre donde el usuario realiza sus elecciones sin ningún tipo de condicionamiento.

Concretamente, durante la navegación tradicional la relación entre dos conceptos no se puede obtener directamente, sólo a través de las relaciones entre sus ítems. De esta forma, se considera que un usuario ha seguido una relación desde el concepto A hasta el concepto B si ha visitado primero un ítem asociado a A y justo después uno asociado a B.

Por otro lado, en SEM-HP únicamente se aplica la regla de aprendizaje de frecuencia, no las de simetría y transitividad. Estas últimas no son necesarias, ya que nuestro objetivo no es enlazar en la navegación los conceptos sino relacionarlos semánticamente.

- La regla de simetría no se aplica porque la existencia de una relación semántica desde A hasta B no significa que exista necesariamente otra relación desde B hasta A. Salvo, claro está, la relación inversa que el autor puede incorporar extendiendo la navegabilidad de ésta.

- La regla de transitividad tampoco se aplica ya que su misión es acortar pasos de navegación y no relacionar conceptos. Esto es, la existencia de una relación entre A y B y otra entre B y C no implica obligatoriamente una relación entre A y C. No obstante, si ésta existiese, el sistema la detectaría al aplicarse la regla de frecuencia cada vez que los usuarios visiten C después de A.

La forma en que se obtiene el peso de cada relación es también sensiblemente diferente en los enfoques comparados. Concretamente, SEM-HP utiliza matrices de transiciones que contabilizan el número de veces que se sigue cada relación y pondera el resultado en función del volumen total de pasos de navegación. Además, este proceso se ve dificultado por la existencia de distintas estructuras y modos de navegación.

Finalmente el tratamiento que se realiza con el resultado obtenido es sensiblemente distinto. En el enfoque de Bollen los pesos se utilizan para ordenar los enlaces que salen de un nodo de mayor a menor “relevancia”. En SEM-HP lo que se pretende es crear/añadir o borrar/ocultar relaciones entre los conceptos de las estructuras conceptuales de memorización y presentación.

Además, en el enfoque descrito, la red hipermedia se reestructura de forma automática a partir de los pesos obtenidos en los enlaces. En este proceso, la intervención del modelador es muy poca o incluso nula. En SEM-HP, por el contrario, el sistema no realiza los cambios identificados, sino que los sugiere al autor para que éste haga efectivos aquellos que considere oportunos (a través del correspondiente proceso de evolución).

6. AHM (PILAR DA SILVA, ET AL.)

6.1 Descripción del trabajo

La arquitectura AHM [DaSilva, 97] [DaSilva, 98] reconoce el concepto de hipermedia adaptativo como la combinación de dos tecnologías muy diferentes: los sistemas tutores inteligentes y los sistemas hipermedia.

AHM se centra en la **adaptación de la navegación**, en concreto en proporcionar soporte a la orientación tanto a nivel local (ayudar al usuario a comprender su posición relativa en el hiperespacio y lo que le rodea) como global (ayudar al usuario a comprender la estructura total del hiperespacio). Para conseguirlo hace uso de dos técnicas de adaptación:

1. Ocultación de enlaces. Se trata de ocultar los enlaces que no conducen a páginas relevantes, ya sea porque no están relacionadas con la meta del usuario o porque el usuario no está listo pata leerlas.
2. Anotación de enlaces. Se aumenta el enlace con un comentario que indica el estado actual del nodo al que apunta.

La primera técnica permite reducir la sobrecarga cognitiva propia de la navegación en sistemas web, debido a que limita el espacio de navegación permitiendo al usuario centrarse sólo en los enlaces más significativos. La segunda técnica proporciona orientación en la medida que ofrece información sobre los nodos disponibles a partir del nodo actual.

En la arquitectura AHM el usuario accede a través de un servidor web que es lanzado por el motor de adaptación. El motor de adaptación accede a la base de datos que contiene el modelo de usuario y el modelo de dominio.

- **Modelo de dominio**:

Existen dos tipos de nodos: conceptos y documentos. Las relaciones pueden ser entre conceptos o entre conceptos y documentos. La relación conceptual más usual es <c1 prerequisito_de c2>. Dicha relación tiene asociado un peso llamado “umbral” que representa el mínimo conocimiento que un usuario debe alcanzar sobre el concepto c1 para acceder al concepto c2.

Entre conceptos y documentos la única relación que se utiliza es <*d* explica *c*> lo que significa que el documento *d* trata sobre el concepto *c* (un documento puede asociarse con más de un concepto). A cada relación de este tipo se le asocia un nivel de dificultad del documento con respecto al concepto.

- **Modelo de usuario y adaptación a la navegación**:

Se inicializa el conocimiento del usuario a cero para todos los conceptos del modelo de dominio. El modelo de usuario se actualiza cada vez que éste visita un documento asociado a un concepto, pero sólo si el documento tiene un nivel de dificultad superior al nivel de conocimiento del concepto actual.

La adaptación de la navegación consiste en presentar al usuario sólo los conceptos relevantes y los documentos apropiados (algoritmo 1). Un concepto es relevante si es un concepto básico, esto es sin prerrequisitos, o todos sus prerrequisitos están satisfechos. Un documento es apropiado si explica un concepto relevante y tiene un nivel de dificultad adecuado para el usuario, es decir no es ni demasiado básico ni demasiado avanzado.

k(s,c) es el conocimiento que el usuario *s* tiene sobre el concepto *c*.

t(c1, c2) es el mínimo conocimiento que debe tener el usuario sobre c1 para poder acceder a c2.

RCs es el conjunto de conceptos relevantes.

RDs es el conjunto de documentos apropiados.

$0 <= k(s,c) <= 100,\ 0 <= t(c1,c2) <= 100.$

RCs = {c / basico(c) or [∀ c' tal que <c' prerrequisito_de c> se cumple $K(s,c') > t(c', c)$] }

RDs = {d / <d explica c> and $c \in$ RCs and [$(K(s,c)-d) \leq$ dificultad (d) $\leq (K(s,c)+d)$] }

Algoritmo 1. Conceptos relevantes y documentos apropiados

Se ha desarrollado un prototipo que sigue el enfoque AHM. Su interfaz de usuario se compone de tres marcos (véase la figura 5):

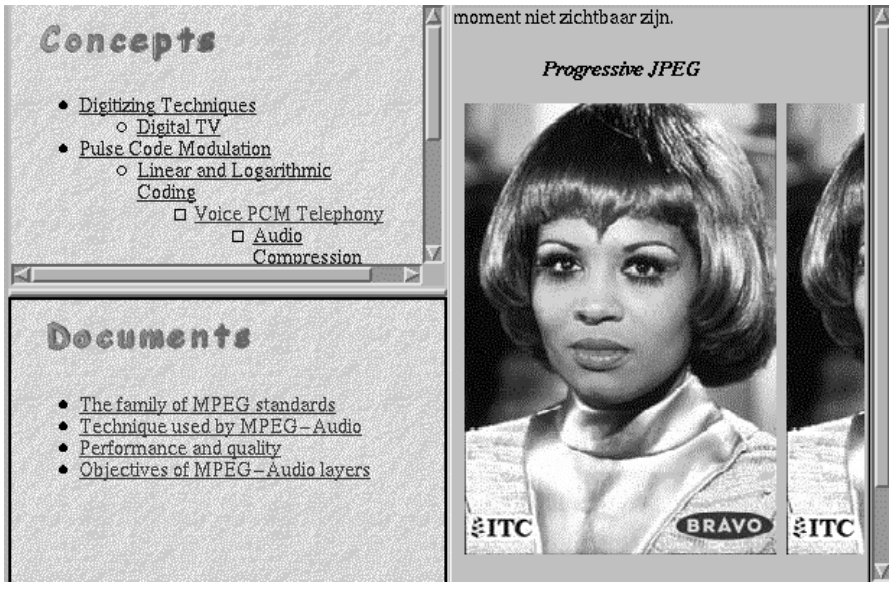


Figura 5: Interfaz AHM

1. El *marco superior izquierdo* presenta la lista de conceptos relevantes.

2. El *marco inferior izquierdo* ofrece la lista de documentos apropiados para el concepto y usuario actual.

3. El *marco derecho* visualiza el contenido del documento seleccionado.

Cada vez que se selecciona un nuevo documento cambia el modelo de usuario, y por lo tanto, los dos marcos de la izquierda tienen que recalcularse. Además se ofrecen tres mapas de conceptos, dos estáticos y uno dinámico. El **mapa dinámico** presenta el concepto actual, sus documentos y sus conceptos vecinos. Uno de los **mapas estáticos** recoge todos los conceptos del dominio y el otro todos los conceptos básicos. Ninguno de los tres mapas es navegable.

En cuanto a los aspectos de implementación cabe destacar que toda la información se almacena en una base de datos Access y luego se traduce dinámicamente a *html*, se utiliza ActiveX Data Objects (ADO) para recuperar información de la BD.

El prototipo ha sido evaluado, a partir de lo cual se han obtenido tres conclusiones fundamentales:

- La interfaz dividida en tres marcos es del agrado de los usuarios.

- La ocultación de enlaces a información ya leída confunde al usuario, prefiriéndose el uso de la anotación en estas situaciones.

- No es suficiente el nivel de dificultad para medir la pericia del usuario sobre un concepto, por lo que se añade un nuevo atributo llamado “cobertura” que indica lo importante que es un documento para la comprensión del concepto que explica.

6.2 Relación con nuestro trabajo

La arquitectura AHM, al igual que SEM-HP, separa conceptos e información en la representación del modelo de dominio. En AHM, todas las unidades de información son documentos. En SEM-HP, en cambio, se trata de trozos de información denominados ítems. De manera que un documento puede ser un ítem o la composición de varios de éstos.

En ambos, el modelo de dominio acepta dos tipos de relaciones: relaciones entre conceptos y relaciones entre conceptos y documentos (ítems en SEM-HP). Sin embargo, en este punto existen algunas diferencias:

- En SEM-HP la asociación entre un concepto y un ítem está etiquetada con un rol que especifica la función de la información que contiene. Así, el usuario tiene una idea más certera del tipo de información que va a leer sobre el concepto: “opinión”, “ejemplo”, “descripción”, “explicación”, etc. En AHM, por simplicidad, la relación entre un concepto y un documento es siempre de tipo “explica”.
- En AHM las relaciones entre los conceptos implican restricciones de navegación, ya que frecuentemente expresan prerrequisitos de conocimiento. En SEM-HP una relación conceptual reflejan únicamente la asociación semántica entre dos conceptos. Los prerrequisitos de conocimiento se expresan fuera del modelo de

dominio, en el Sistema de Aprendizaje. De esta forma no se fuerza la representación del conocimiento para conseguir una determinada navegación, y se permiten distintas posibilidades de navegación sobre un mismo modelo de dominio.

- En AHM los prerrequisitos pedagógicos son establecidos a nivel conceptual mientras que en SEM-HP se establecen a nivel de ítem, lo cual permite al autor mayor expresividad. Ya que, un prerrequisito entre dos conceptos puede ser emulado mediante prerrequisitos entre sus ítems, pero no al contrario.

Por ejemplo <c1 prerrequisito_de c2> puede simularse como <d11 and d12 prerrequisito_de d2i> donde d11 y d12 son los documentos disponibles sobre el concepto c1 y d2i se instancia con cualquier documento asociado a c2.

Sin embargo, con prerrequisitos entre conceptos no es posible simular la situación en que dos documentos asociados a un mismo concepto tienen prerrequisitos diferentes. Por ejemplo: <d11 and d12 prerrequisito_de d21> y <d31 prerrequisito_de d22>.

En SEM-HP, una vez que el usuario satisface todos los requisitos de conocimiento necesarios para comprender un ítem, la dificultad o nivel de especialización de éste únicamente va a repercutir en el tiempo que se necesita para asimilar su contenido. En AHM, debido a que los requisitos de conocimiento se imponen a nivel conceptual, se hace necesario un mecanismo adicional que permita diferenciar entre los documentos asociados a un mismo concepto. Dicho mecanismo evalúa la dificultad del documento para decidir si éste es o no apropiado para el usuario.

Respecto al modo en que se obtiene el conocimiento del usuario sobre los conceptos y documentos, ambos modelos son muy diferentes. En AHM el conocimiento del usuario sobre un documento se supone “total” después de su lectura. Y el conocimiento sobre un concepto se establece como el mayor nivel de dificultad de un documento asociado a dicho concepto que el usuario haya leído. En cierto modo, se está mezclando dificultad y conocimiento, dos conceptos que nosotros consideramos distintos. Además, no porque un ítem sea más complicado tiene que ser más relevante para entender el concepto.

En SEM-HP, después de que el usuario lee un ítem, el sistema actualiza su conocimiento sobre éste y, quizá, sobre otros ítems que tratan información similar. Esta actualización se hace de acuerdo a la regla que el autor ha asociado al ítem visitado, y permite incrementar el conocimiento de cada ítem de forma diferente, según la importancia que respecto a éste tiene la lectura realizada. Después, el conocimiento sobre un concepto se calcula en función de cuánto conoce el usuario acerca de cada uno de sus ítems, ponderando el conocimiento de cada uno según lo relevante que sea para comprender el concepto. La influencia de cada ítem es indicada por el autor en la regla de peso definida para el concepto, y coincide con la noción de “cobertura” concluida durante la evaluación de AHM.

Respecto a la navegación, en AHM, al igual que en SEM-HP, se opta por una interfaz dividida en marcos. La interfaz de AHM proporciona como soporte de orientación un conjunto de mapas que muestran, desde distintas perspectivas, las relaciones existentes entre los conceptos. Sin embargo, esos mapas no son navegables, sólo informativos. En SEM-HP la navegación se realiza directamente sobre una red semántica que visualiza las relaciones conceptuales y funcionales definidas en el subdominio de conocimiento navegado.

Los usuarios de AHM perciben el conjunto de documentos apropiados como una lista que aparece cuando seleccionan uno de los conceptos relevantes que se le facilitan en otra lista. En SEM-HP, el usuario distingue visualmente sobre la red de navegación si un ítem es inaccesible (borde casi transparente), accesible pero no idóneo (borde negro discontinuo) o idóneo (borde negro). De esta forma, se guía la navegación del usuario sin desproveerle de una visión de los ítems global (no sólo para un concepto determinado) y completa (no sólo los ítems apropiados, sino todos, anotados según su condición actual).

7. CONSIDERACIONES GENERALES

En general, al comparar nuestro trabajo con todos los que aquí se presentan, y con el resto de trabajos revisados, encontramos las siguientes características que hacen interesante nuestra propuesta:

1. **Proceso de desarrollo incremental y en fases**:

Son realmente muy pocos los autores que definen un proceso de desarrollo específico para sistemas hipermedia adaptativos. En nuestro caso, consideramos que un proceso de desarrollo establecido en fases e incremental es esencial para conseguir la mayor funcionalidad adaptativa en los sistemas resultantes.

Concretamente, el proceso propuesto se divide en cuatro fases, que separan perfectamente los aspectos de representación, presentación, navegación y adaptación propios de este tipo de sistemas. Esta separación de aspectos permite que las fases sean incrementales. Esto es, a partir de la representación del dominio de conocimiento realizada en la fase inicial, el autor puede crear varias presentaciones durante la segunda fase. Del mismo modo, durante las fases tercera y cuarta, el autor puede definir distintos esquemas de navegación y adaptación para cada una de estas presentaciones. De esta forma se multiplican las estructuras de aprendizaje disponibles finalmente, lo que capacita al sistema para satisfacer mejor la pluralidad de sus usuarios (capítulo 14, Elección de la EC_A).

2. **Capacidad Evolutiva**:

Prácticamente ninguno de los modelos, arquitecturas o sistemas que encontramos en la literatura han sido dotados de un mecanismo evolutivo, que permita al autor modificar las representaciones realizadas en el modelo de dominio o los formalismos que proporcionan la adaptación del sistema.

Sin embargo, en SEM-HP el autor dispone de mecanismos de cambio que le permiten modificar el sistema para mejorarlo. Estas acciones evolutivas garantizan la integridad del sistema tras cada modificación, impidiendo la ejecución de cambios inconsistentes y efectuando la propagación del cambio en aquellas situaciones que lo requieran.

Concretamente, en el Sistema de Aprendizaje existen acciones evolutivas para modificar cualquier elemento usado para generar la adaptación: reglas de peso, reglas de conocimiento, reglas de actualización, etiquetado de las estructuras de navegación y modelo de usuario.

La evolución permite mejorar todo el proceso de construcción del sistema hipermedia, especialmente su mantenimiento. Pero, además, al garantizarse la integridad durante el proceso de cambio se potencia la usabilidad y reusabilidad del sistema

En la tabla 1 se muestran los modelos de evolución proporcionados en cada uno de los modelos de adaptación revisados. Observe que todos ellos presentan únicamente el sexto modelo, que se corresponde con la auto-adaptación funcional del sistema. Por lo tanto, se centran en la capacidad de adaptarse a sus usuarios pero no proporcionan el soporte necesario para que el autor pueda hacer evolucionar consistentemente la estructura del sistema software. Por el contrario, en SEM-HP cuatro de los seis modelos evolutivos están presentes (capítulo 23, Consideraciones sobre evolución y adaptación).

Tabla 1: Caracterización evolutiva de los trabajos relacionados

		Modelos de Evolución					
		1	**2**	**3**	**4**	**5**	**6**
Modelo de adaptación	**SEM-HP**	א	א			א	א
	ADAPTS						א
	PUSH						א
	AHAM						א
	AHM						א

3. **Diversidad de modos de navegación**:

Los modelos de adaptación estudiados se centran en modificar la forma y funcionalidad de la estructura de navegación proporcionada, pero son muy pocos los que admiten distintos modos de recorrerla. Y los que lo hacen, se limitan a ofrecer estructuras de navegación alternativas, sobre las que, en realidad, se navega con la misma filosofía.

En SEM-HP, por el contrario, se ofrecen cuatro modos de navegación conceptualmente diferentes, que varían en la estructura de navegación utilizada, los prerrequisitos de navegación, la unidad de navegación, y las técnicas de adaptación aplicadas. Estos modos representan filosofías distintas para recorrer la información, pudiendo el usuario elegir entre:

a) Navegación libre o guiada por el sistema.

b) Navegación coherente con las relaciones semánticas existentes entre los conceptos o restringida en función de su conocimiento.

c) Navegación de la información o de los conceptos.

A pesar de las diferencias que existen entre los cuatro modos de navegación ofertados, la adquisición de conocimiento se trata de modo uniforme en todos ellos: el sistema ejecuta la regla de actualización de un ítem después de que el usuario lo visite en posesión de los prerrequisitos de conocimiento necesarios para asimilar su contenido. Esta gestión solidaria del modelo de usuario, unida al hecho de que todos los modos

comparten una representación de conocimiento común (la EC_M), hace que se trate de modos alternativos pero no inconexos.

4. **Capacidad adaptativa**:

Respecto a la capacidad adaptativa, la mayoría de los trabajos revisados se centran en unas cuantas técnicas de adaptación. Nuestra propuesta, sin embargo, es capaz de integrar y llevar a cabo, modos de adaptación al usuario muy diferentes. Empezando, por elegir de forma personalizada la estructura de navegación y las reglas de aprendizaje. Y terminando con la generación de rutas orientadas a la meta del usuario, a medida de su estado de conocimiento y preferencias.

Además, el sistema aplica unas técnicas u otras dependiendo, no sólo de las características del usuario, sino también del modo de navegación que éste ejercita. Esto permite ajustar la adaptación en función del modo con que el usuario ha decidido recorrer la información, realizando únicamente aquellas técnicas que pueden ayudarle durante la filosofía de navegación elegida.

La aplicación de esta variedad de técnicas es posible gracias a la utilización de potentes y flexibles formalismos lógicos como las reglas de conocimiento, de peso y de actualización, a la completitud de un modelo de usuario de grano fino capaz de recoger toda la información necesaria, así como a la gestión adaptable y adaptativa de éste.

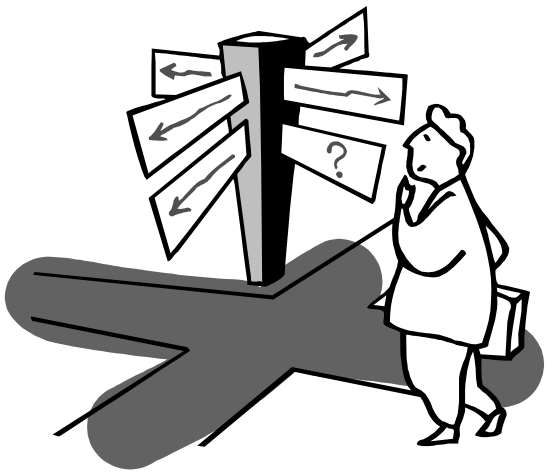
Para concluir, se presenta en la tabla 2 la caracterización adaptativa de los trabajos revisados, de acuerdo a la taxonomía de sistemas hipermedia adaptativos del capítulo 3. Se incluye, también, la caracterización del modelo SEM-HP (capítulo 23) para facilitar la comparación.

Tabla 2: Caracterización adaptativa de los trabajos relacionados

	SEM-HP	**ADAPTS**	**PUSH**	**AHAM**	**AHM**
Aplicabilidad	General	Específico	Específico	General	General
Adaptación a	Usuario individual y grupo	Usuario individual (técnico)	Usuario individual	Usuario individual	Usuario individual
Adaptación de	Sobre todo Navegación	Navegación y Presentación	Sobre todo Presentación	Navegación y Presentación	Sobre todo Navegación
Prerrequisitos	Pedagógicos y de orden	Pedagógicos	De orden	Pedagógicos	Pedagógicos
Origen de la información	Abierto	Indexación	Base de Datos	Abierto	Base de Datos
Interacción usuario-adaptación	Adaptativo y Adaptable	Adaptativo	Adaptativo y Adaptable	Adaptativo	Adaptativo
Creación de documentos	Dinámico en resúmenes	Estático	Dinámico	Estático	Traducción dinámica
Información contextual	Ninguna	_	Ninguna	Ninguna	Ninguna

CAPÍTULO 25

Conclusiones y Trabajo Futuro



Resumen

En este capítulo se realiza una justificación general de la consecución de los objetivos formulados en el proyecto de tesis. Así como una breve recapitulación de todos los resultados obtenidos durante el desarrollo de la tesis doctoral: aportaciones que avalan el logro de los objetivos y publicaciones que respaldan la utilidad de dichas aportaciones. Finalmente se esbozan las principales líneas de trabajo planteadas para un futuro próximo, y algunas líneas en las que ya hemos empezado a trabajar.

Tabla de contenidos

Conclusiones y Trabajo Futuro

1. INTRODUCCIÓN

En la presente tesis se ha propuesto un modelo de adaptación al usuario que permite desarrollar, como era su objetivo, **sistemas hipermedia adaptativos, integrales y evolutivos**. Esto es:

- Sistemas hipermedia capaces de ajustarse al máximo a las características de sus usuarios,

- dotados, además, de los mecanismos necesarios para que el autor pueda realizar un proceso de evolución consistente siempre que así se requiera.

De acuerdo a los objetivos planteados inicialmente, las características finales del modelo que permiten calificar la **adaptación** realizada de **integral** son las siguientes:

a) Aplicabilidad general respecto al dominio de conocimiento y uso del sistema.

b) Disponibilidad de gran variedad de métodos y técnicas de adaptación que permiten ajustar la funcionalidad del sistema a cada usuario.

c) Uso de prerrequisitos de orden y conocimiento.

d) Abierto ante la incorporación de nuevos ítems de información.

e) Gestión adaptable y adaptativa del modelo de usuario.

f) Generación automática de documentos mediante composición de ítems.

g) Capacidad para adaptar las estructuras de representación a la navegación colectiva de los usuarios.

Los elementos básicos del modelo que se utilizan para dar soporte a las anteriores características y capacidades son: el modelo de usuario, las reglas de conocimiento, las reglas de actualización, las reglas de peso, y el etiquetado de las estructuras de aprendizaje. Sobre los anteriores se superponen un conjunto de elementos adicionales (matriz de transiciones, estructura de composición de los resúmenes, árbol de ítems deseables,...), así como un conjunto de técnicas, procedimientos y algoritmos que permiten la adaptación integral enumerada más arriba.

Respecto a los **requisitos evolutivos**:

a) Se proporciona al autor un conjunto de acciones para realizar, en el momento que considere oportuno, modificaciones sobre los elementos del modelo.

b) Se incorporan restricciones para verificar que los cambios introducidos son consistentes.

c) Se definen mecanismos para evaluar y propagar la repercusión del cambio dentro del sistema donde se integra.

Cada uno de los elementos del modelo ha sido convenientemente especificado y formalizado. Planteando, desde un principio su integración como Sistema de Aprendizaje en SEM-HP [García, 01c]. Esto ha permitido que el modelo de adaptación sea construido sobre bases tan sólidas y valiosas como las siguientes:

- Estructura sistémica que permite separar los aspectos de conocimiento (memorización y presentación), navegación y aprendizaje.

- Representación explícita de la semántica en una red hipermedia donde se distinguen dos tipos de nodos: conceptos e ítems.

- Arquitectura de dos niveles para la evolución: sistema y meta-sistema.

Para conseguir una integración completa se ha estudiado la interacción del Sistema de Aprendizaje, a nivel de estructura, funcionamiento y evolución, con el resto de sistemas que constituyen SEM-HP. Definiendo, en consecuencia, un mecanismo de propagación externa capaz de mantener en estado consistente los elementos del Sistema de Aprendizaje tras la modificación de cualquier elemento en otro sistema de SEM-HP.

Finalmente, se ha desarrollado una herramienta de prototipado, que permite valorar la aplicación práctica de algunas de las características más relevantes del modelo.

2. RECAPITULACIÓN DE RESULTADOS

Los resultados obtenidos durante estos años de trabajo pueden dividirse en dos tipos:

i) Resultados abstractos, y

ii) Resultados tangibles.

Los **resultados abstractos** (i) se componen de un cúmulo de ideas adquiridas y perfeccionadas durante el proceso de revisión, análisis e investigación llevado a cabo en torno al área de los sistemas hipermedia adaptativos. Así, como de un conjunto de experiencias relacionadas que me ha permitido un crecimiento científico y personal.

El conocimiento adquirido, fruto de no pocas reflexiones, ha sido aplicado y realimentado durante el desarrollo del modelo de adaptación que aquí se materializa. Por lo tanto, se pueden considerar las aportaciones descritas y formalizadas en la presente tesis doctoral como el principal componente de los **resultados tangibles** (ii).

Junto al contenido de la tesis, enumerado en la sección 2.1, cabe destacar como resultado tangible las diversas publicaciones que nos han permitido dar a conocer nuestro trabajo dentro de la comunidad científica. Estas publicaciones se muestran en la sección 2.2 clasificadas según su ámbito sea nacional o internacional, e incluyen actas de congresos, capítulos de libro y artículos de revista.

2.1 Contenido de la tesis

1) Se ha estudiado el modelo SEM-HP desarrollado por la doctora García-Cabrera en su tesis doctoral [García, 01c].

- Se ha comprendido y analizado el modelo, y

- Se ha identificado el trabajo pendiente para completarlo: Sistema de Aprendizaje, propagación externa del cambio y retroalimentación adaptativa.

2) Se ha realizado un estudio bibliográfico de los trabajos desarrollados en el área de los sistemas hipermedia en general, y de los sistemas hipermedia adaptativos en particular.

    - Se han extraído las ventajas e inconvenientes observadas en los sistemas hipermedia adaptativos,

    - Se han identificado las características comunes en la mayoría de estos sistemas, y

    - Se ha desarrollado una taxonomía que permite caracterizar cualquier sistema hipermedia adaptativo de acuerdo a ocho criterios de clasificación diferentes.

3) Se ha realizado un estudio conceptual e histórico de los principales elementos que pueden aparecer en el entorno de un sistema hipermedia adaptativo.

    - Sistemas educaciones inteligentes y adaptativos

    - Sistemas tutores inteligentes

    - Web semántica: XML, RDF, ontologías, etc.

4) Se han revisado las principales teorías filosóficas del conocimiento, y se ha enmarcado nuestra visión epistemológica como coherentista y constructiva.

5) Se ha reflexionado sobre las distintas metodologías que permiten la evolución del software.

    - Se han recopilado y comparado distintas definiciones de los términos “sistema” y “modelo”.

    - Se han estudiado dos taxonomías de evolución, desarrolladas por Massimo Felici [Felici, 03] y por Mens et. al. [Mens, 03] respectivamente.

    - Se han analizado las carencias de los métodos de evolución existentes, decidiéndonos como base evolutiva de nuestro trabajo por el modelo MEDES [Parets, 95].

6) Se han determinado las características deseables para el Sistema de Aprendizaje, estableciendo a partir de éstas los objetivos de la tesis, y planteando desde el principio la integración del trabajo realizado dentro del modelo SEM-HP.

7) Se ha establecido una representación formal para el estado de conocimiento del usuario, basada en la utilización de etiquetas semánticas para expresar grados de conocimiento.

8) Se ha definido formalmente un lenguaje lógico de autor, Rk-autor, que permite expresar con bastante flexibilidad las condiciones de conocimiento óptimas para la visita de un ítem.

   - Se ha especificado formalmente la sintaxis y la semántica del lenguaje.
   - Se ha demostrado la completitud y consistencia del lenguaje.
   - Se ha propuesto una interfaz de autor apropiada.

9) Se ha definido formalmente un lenguaje lógico para representar internamente las reglas de conocimiento, Rk. Esta representación interna permite separar las condiciones para visitar un ítem en restricciones de accesibilidad y restricciones de idoneidad.

   - Se ha especificado formalmente la sintaxis y la semántica del lenguaje Rk.
   - Se ha demostrado la equivalencia entre Rk y Rk-autor.
   - Se ha establecido un mecanismo de traducción automática de Rk-autor a Rk.

10) Se ha especificado convenientemente (argumentos, definición, pre-condiciones, pos-condiciones, efecto, salida, propagación,...) un conjunto de acciones evolutivas que permiten al autor añadir, eliminar o modificar las reglas de conocimiento en su sintaxis de autor, realizándose automáticamente la propagación que corresponda a la representación interna de éstas.

11) Se ha descrito formalmente un mecanismo lógico, denominado reglas de actualización (Ru), que sirve al autor para especificar la adquisición de conocimiento después de cada visita del usuario a un ítem.

   - Se ha especificado formalmente la sintaxis y semántica del lenguaje Ru.
   - La estructura general de una regla Ru ha sido definida para que permita actualizar varios ítems como consecuencia de una visita.
   - Se han definido predicados para que la actualización de un ítem pueda ser: fija o incremental, relativa o absoluta, cada vez o sólo tras la primera visita.
   - Se ha definido un mecanismo para generar el conjunto de reglas de actualización por defecto.
   - Se ha propuesto una interfaz de autor apropiada.

12) Se han especificado convenientemente un conjunto de acciones evolutivas para que el autor pueda modificar los predicados de actualización del cuerpo de una regla. Y para que el sistema pueda crear y eliminar las reglas de actualización que corresponda.

13) Se ha estudiado la consistencia de los formalismos propuestos, planteando dos procedimientos para comprobar si el conjunto de reglas de conocimiento por si sólo o en conjunción con las reglas de actualización asegura la inexistencia de ítems

inalcanzables, esto es, ítems que nunca van a ser accesibles para el usuario sea cual sea la navegación (por conocimiento) que realice.

- Se ha incorporado la aplicación de estos algoritmos a las acciones evolutivas donde se requiere para garantizar que tras el cambio realizado se sigue satisfaciendo la propiedad de alcanzabilidad.

- Se ha calculado la complejidad de ambos algoritmos y se ha justificado frente a la complejidad, mucho mayor, que se obtendría si se hace la comprobación por fuerza fruta, esto es, generando todo el árbol de navegación.

14) Se ha definido formalmente una función matemática denominada regla de peso (Rw) que permite obtener el grado de conocimiento sobre un concepto a partir del grado de conocimiento que el usuario posee sobre cada uno de sus ítems asociados.

    - Se ha definido la estructura de la regla de peso de forma que el autor pueda expresar la importancia de cada ítem ponderando su conocimiento para el cálculo del conocimiento del concepto.

    - Se ha establecido un mecanismo por el cual el sistema es capaz de generar la regla de peso por defecto para cada concepto.

15) Se han definido convenientemente las acciones evolutivas necesarias para que el autor pueda modificar la distribución de pesos de una regla de peso. Y, para que el sistema pueda crear o eliminar reglas de peso y añadir términos en éstas cada vez que se requiera.

16) Se ha identificado la información que el sistema debe conocer acerca del usuario para poder realizar posteriormente la adaptación deseada.

    - Para que la adaptación sea lo más completa posible, se ha incluido en el modelo de usuario:

        + Datos personales.
        + Grado de conocimiento y visitas realizadas a cada ítem y concepto.
        + Preferencias del usuario respecto a las características de los ítems, la longitud de las rutas guiadas o la estructura de composición de los documentos resumen.
        + Experiencia de navegación y en la materia.
        + Subdominio de conocimiento que le interesa, ítems concretos en cuya visita está más interesado y meta de conocimiento que desea alcanzar.

    - Para cada elemento del modelo de usuario se ha descrito su significado, utilidad y representación. Además, de la forma en que éste es inicializado y actualizado.

    - A partir de lo anterior, se han definido y especificado formalmente el esquema y la instancia del modelo de usuario.

- Y se ha propuesto una interfaz de interacción apropiada para el usuario.

17) Se han especificado convenientemente las acciones evolutivas que el sistema va a ejecutar para añadir, eliminar o modificar entradas en el modelo de usuario cada vez que sea necesario.

18) Con objeto de integrar plenamente el Sistema de Aprendizaje dentro del modelo SEM-HP se han investigado los cambios externos que le afectan, detectando la existencia de dos secuencias de propagación diferentes: desde el Sistema de Memorización y desde el Sistema de Presentación.

    - Se han identificado los elementos concretos del Sistema de Aprendizaje que pueden verse afectados por un cambio realizado en el Sistema de Memorización o de Presentación, y

    - Se ha detallado el proceso de propagación necesario, esto es las acciones evolutivas del Sistema de Aprendizaje que hay que ejecutar tras el cambio, para garantizar la consistencia global del modelo.

19) Se ha definido una estructura de etiquetado haciendo uso de la cual el autor puede especificar para cada estructura de aprendizaje (EC_A) el perfil de usuario para el que ésta es adecuada.

    - Se ha precisado el significado y los valores posibles para cada una de las tres etiquetas asociadas a una EC_A.

    - Se han definido las acciones evolutivas necesarias para que el autor pueda modificar el etiquetado de una $EC_{A.}$

    - Se ha desarrollado un procedimiento mediante el cual el sistema es capaz de elegir automáticamente la EC_A más afín a un usuario de acuerdo a su experiencia y subdominio de interés.

20) Se han definido cuatro modos de navegación diferentes: tradicional, por conceptos, por relación conceptual y por conocimiento; para que el usuario pueda elegir el que más le convenga en cada momento.

    - Se han analizado sus ventajas y se ha optado por realizar la navegación directamente sobre la red semántica en todos los modos ofertados.

    - Se ha establecido un mecanismo de adquisición de conocimiento uniforme para todos los modos de navegación, basado en las reglas de actualización y de conocimiento.
    - Se han asignado unas reglas de navegación propias para cada modo:
        + Navegación tradicional: selección libre de ítems.
        + Navegación por conceptos: selección libre de conceptos.
        + Navegación por relación conceptual: selección de ítems restringida por las reglas de orden.

+ Navegación por conocimiento: selección de ítems restringida por las reglas de conocimiento.

- Se ha establecido un conjunto de técnicas de adaptación específico para cada modo de navegación:
    + Navegación tradicional: anotación de los ítems previamente visitados.
    + Navegación por conceptos: anotación del número de visitas sobre cada concepto y personalización de la estructura de los resúmenes (texto desplegable y reordenación de secciones).
    + Navegación por relación conceptual: anotación del número de visitas sobre cada ítem, anotación positiva del concepto actual y las relaciones que parten de él, anotación positiva de los conceptos disponibles y los ítems accesibles, y anotación negativa de ítems inaccesibles.
    + Navegación por conocimiento: anotación de ítems visitados, anotación de conocimiento sobre ítems y conceptos, anotación negativa de ítems no idóneos, ocultación y deshabilitación de ítems no accesibles, anotación positiva de ítems deseables y generación de rutas guiadas.

21) Se ha especificado un mecanismo dinámico que permite generar automáticamente el documento resumen de un concepto cuando éste es seleccionado durante la navegación por conceptos.

    - Se ha definido una estructura de composición de resúmenes genérica, y

    - Se han descrito los mecanismos necesarios para que el autor defina inicialmente dicha estructura y para que el usuario la personalice después.

22) Se ha definido un mecanismo de retroalimentación adaptativa capaz de sugerir al autor modificaciones en las estructuras conceptuales de presentación y memorización, para que éstas converjan hacia la concepción mental que de ellas tienen los usuarios del sistema.

    - Se han definido formalmente matrices de transiciones destinadas a almacenar las asociaciones conceptuales seguidas durante la navegación libre (tradicional o por conceptos) de los usuarios.

    - Se han propuesto mecanismos de análisis y comparación para obtener: relaciones imposibles y conceptos olvidados, perfil de las presentaciones más y menos visitadas, relaciones conceptuales que faltan o sobran en las estructuras de autor, etc.

23) Se han establecido los mecanismos necesarios para realizar un proceso de navegación dirigido mediante las reglas de orden previamente definidas en [García, 01c].

    - Se ha definido el concepto de *starting-point* o punto de partida de la navegación.

- Se han especificado los procedimientos necesarios para detectar los conceptos disponibles e ítems accesibles.

24) Se ha establecido el procedimiento a seguir para determinar automáticamente la accesibilidad e idoneidad de un ítem en función de las reglas de conocimiento definidas por el autor para la visita de éste y el estado de conocimiento que actualmente posee el usuario.

25) Se ha definido el concepto de ítem deseable como un ítem accesible cuya visita acerca al usuario a la consecución de sus intereses y/o meta de conocimiento.

    - Se han descrito y especificado los algoritmos necesarios para construir un árbol denominado TreeD, en cuyo último nivel se hayan los ítems deseables.

    - En cada caso, se ha establecido la forma en que el árbol debe ser actualizado durante la navegación del usuario.

26) Se ha estudiado la conveniencia de usar una técnica de consejo global. Y se han propuesto, en consecuencia, los mecanismos necesarios para la generación automática de rutas que guíen al usuario hacia la consecución de su meta, intentando que la longitud de la ruta y los ítems incluidos en la misma se ajusten en el mayor grado posible a sus preferencias.

    - Se ha realizado una especificación formal del problema y del árbol de búsqueda.

    - Se ha realizado un análisis de diferentes estrategias de búsqueda, tanto ciega como respaldada con información.

        + Se ha estudiado en cada caso la complejidad computacional, y

        + Se han propuesto algunas soluciones para mejorarla. Entre ellas, dos heurísticas para la búsqueda A*.

    - Se ha propuesto y definido un algoritmo Greedy.

27) Se ha analizado la relación existente entre la adaptación al usuario y la evolución del software, concluyendo que la segunda incluye a la primera.

28) Se ha realizado una caracterización del modelo SEM-HP desde dos puntos de vista, el evolutivo y el adaptativo.

29) Se ha llevado a cabo un estudio comparativo de nuestro trabajo con otros trabajos afines, extrayendo en cada caso las similitudes y diferencias, y proporcionando una justificación general de los aspectos más destacables de nuestro modelo.

2.2 Publicaciones realizadas

A continuación se proporciona un listado de las publicaciones realizadas. La tabla 1 en la sección 2.2.1 muestra las publicaciones de ámbito nacional y la tabla 2 en la sección 2.2.2 las de ámbito internacional.

2.2.1 Publicaciones de ámbito nacional

Tabla 1: Publicaciones de ámbito nacional

Medina Medina, N.; García Cabrera, L.; Rodríguez Fórtiz, M.J.; Parets Llorca, J. "Adaptación al Usuario en Sistemas Hipermedia: El Modelo SEM-HP". *II Jornadas de trabajo DOLMEN*. Pp: 175-185. Valencia, 12 y 13 de Marzo de 2002.
Medina Medina, N.; Molina Ortiz, F.; García Cabrera, L.; Rodríguez Fortiz, M.J. "Un Modelo para el Desarrollo de Sistemas Hipermedia Adaptativos". *II Workshop de Investigación sobre nuevos paradigmas de interacción en entornos colaborativos. COLINE'02.* Granada, 11 y 12 de Noviembre de 2002.
Medina Medina, N.; García Cabrera, L.; Parets Llorca, J. "Taxonomía de Sistemas Hipermedia Adaptativos". *Taller en sistemas hipermedia colaborativos y adaptativos*. VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial, 18 de Noviembre de 2002.
Medina Medina, N.; García Cabrera, L. "Modelos de Evolución en SEM-HP". *III Jornadas de trabajo DOLMEN*. Pp: 71-77. VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial, 18 de Noviembre de 2002.
Molina Ortiz, F.; García Cabrera, L.; Medina Medina, N.; Hurtado Torres, MV. "Editor de Estructuras Conceptuales Evolutivas: Consideraciones Prácticas". *III Jornadas de trabajo DOLMEN*. Pp: 77-83. VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial, 18 de Noviembre de 2002.
Medina Medina, N.; García Cabrera, L.; Molina Ortiz, F. "Diversidad de Tipos de Navegación en un SHA". *Taller en sistemas hipermedia colaborativos y adaptativos (2ª edición)*. Pp: 1-9. VIII Jornadas de Ingeniería del Software y Bases de Datos. Alicante, del 12 al 14 de Noviembre de 2003.
Molina Ortiz, F.; Medina Medina, N.; García Cabrera, L. "JSEM-HP, una herramienta de autor para el desarrollo de sistemas hipermedia adaptativos evolutivos". *IV Jornadas de trabajo DOLMEN*. Pp: 110-115. VIII Jornadas de Ingeniería del Software y Bases de Datos. Alicante, 11 de Noviembre de 2003.

2.2.2 Publicaciones de ámbito internacional

Tabla 2: Publicaciones de ámbito internacional

Medina Medina, N.; García Cabrera, L.; Torres Carbonell, J.J.; Parets Llorca, J. "Evolution in Adaptative Hypermedia Systems". *Proceedings of the first international workshop on principles of software evolution. IWPSE 2002*. Pp: 34-38. Orlando, Florida, USA. May 19-20, 2002.
Medina Medina, N.; García Cabrera, L.; Rodríguez Fortiz, M.J.; Parets Llorca, J. "Adaptation in an Evolutionary Hyperpedia System: Using Semantic and Petri Nets". *Adaptative Hypermedia and Adaptative Web-Based Systems (second international conference, AH 2002).* Málaga, Spain. May 29-31, 2002. *Lectures Notes in Computer Science.* Vol. 2347. ISSN: 0302-9743. Pp: 284-295. LNCS Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany.
Medina Medina, N; Molina Ortiz, F.; García Cabrera, L; Parets Llorca, J. "Personalized Guided Routes in an Adaptive Evolutionary Hipermedia System". *Proceedings of the 9th International workshop on computer aided system theory". Eurocast'03.* Extended Abstract, Pp: 135-139. Las Palmas de Gran Canaria, Spain. February 24-28, 2003.
Medina Medina, N; Molina Ortiz, F.; García Cabrera, L; Parets Llorca, J. "Personalized Guided Routes in an Adaptive Evolutionary Hipermedia System". 9th International workshop on computer aided system theory". Eurocast'03*. Selected Paper in Lectures Notes in Computer Science*. Vol. 2809. ISSN: 0302-9743. ISBN: 3-540-20221-8. Pp: 196-207. LNCS Editorial Springer-Verlag. Berlin Heidelberg, New York.

Medina Medina, N.; Molina Ortiz, F.; Rodríguez Fortiz, M.J.; García Cabrera, L. "An Architecture for the Development, Evolution and Adaptation of Hipermedia Systems". *Proceedings of the international conference on software engineering research and practice. SERP'03*. Volumen I. ISBN:1-932415-19-X. Pp:112-119. Las Vegas, Nevada, USA. June 23-26, 2003.
Paderewsky Rodríguez, P.; Torres Carbonell, J.; Rodríguez Fórtiz, M.J.; Medina Medina, N.; Molina Ortiz, F. "A Software System Evolutionary and Adaptive Framework. Application to Agent Based Systems". *Proceedings of the international conference on software engineering research and practice. SERP'03*. Volumen I. ISBN:1-932415-19-X. Pp: 142-149. Las Vegas, Nevada, USA. June 23-26, 2003.
Paderewsky Rodríguez, P.; Torres Carbonell, J.; Rodríguez Fórtiz, M.J.; Medina Medina, N.; Molina Ortiz, F. "A Software System Evolutionary and Adaptive Framework. Application to Agent Based Systems". *Selected Paper in Journal of systems architecture 50 (2004).* Pp: 407-416. Available online at www.sciencedirect.com. Editorial Elsevier.
Medina Medina, N; Molina Ortiz, F.; García Cabrera, L; Rodríguez Fórtiz, M.J. "Coevolution of Models of an Adaptive Hipermedia System". *Proceedings of the 7th biennial world conference on integrated design and process technology. IDPT'03.* ISSN: 1090-9389. Volumen II. Pp: 18, 38-44. Austin, Texas, USA. December 3-6, 2003.
Martín Puente, E; Medina Medina, N; García Cabrera, L. "Educational Support in an Adaptive Hipermedia System: SEM-HP model". *Proceedings of the 2nd international conference on multimedia ICT's in Education. M-ICTE'03*. Volumen III. ISBN: 84-96212-12-2. Pp: 1642-1646. Badajoz, Spain. December 3-6, 2003.
Medina Medina, N; Molina Ortiz, F.; García Cabrera, L. "A Hypermedia Model for an Adaptive Learning". *Proceedings of the IADIS international conference. e-Society 2004.* Volumen I. ISBN: 972-98947-5-2. Pp: 276-283. Ávila, Spain. July 16-19, 2004.

3. TRABAJO FUTURO

Una vez recopilados los resultados, y observando el trabajo realizado desde un punto de vista de conjunto, es momento de plantearse mejoras y ampliaciones viables. No cabe pues dar carpetazo sin más, es necesario mirar al futuro y plantearse nuevas metas y, ¿por qué no?, apostar por las posibilidades del trabajo realizado.

Los principales objetivos que proyectamos abarcar en un futuro más o menos inmediato se describen en las subsecciones siguientes y son: trabajar en la implementación del modelo, de modo que tras el ciclo de prototipado incremental obtengamos una herramienta de autor final y completa (sección 3.1), aumentar las posibilidades educacionales del modelo (sección 3.2) e incluir la colaboración como parte del mismo (sección 3.3).

También, se pretende estudiar la integración de nuestra propuesta dentro de la web semántica. En este sentido nos planteamos evaluar las ventajas e inconvenientes de aplicar la tecnología de la web semántica para representar las estructuras conceptuales del modelo SEM-HP, anticipando las dificultades derivadas del carácter no evolutivo de lenguajes de representación de ontologías como RDF y OWL.

3.1 Implementación y validación

Completar la implementación de la herramienta JSEM-HP es nuestro objetivo más inmediato. En este sentido son varias las tareas pendientes:

a) Implementar el procedimiento que identifica los ítems deseables a partir de una meta o un conjunto de intereses.

b) Implementar el modo de navegación por conceptos. En particular, el mecanismo de composición dinámico para crear y personalizar los resúmenes.

c) Implementar de acuerdo a lo especificado la técnica de retroalimentación adaptativa.

d) Depurar la implementación que actualmente se tiene del mecanismo para identificar conceptos disponibles e ítems accesibles en la navegación por relación conceptual.

e) Hacer operativo el mecanismo de generación de rutas guiadas. En la actualidad está implementado el algoritmo A* con la heurística h1, pero faltan aspectos de interfaz para mostrar las rutas generadas y en el modelo de usuario para solicitar las preferencias de éste.

En relación con la tarea e), aunque se han realizado ya algunos experimentos que manifiestan la utilidad de las heurísticas propuestas, se desea ampliar el estudio empírico para comparar estadísticamente las estrategias de búsqueda planteadas, midiendo la calidad de las rutas obtenidas y el tiempo de respuesta.

Algunas de las anteriores tareas están actualmente en proceso y se prevé poco esfuerzo para finalizarlas completamente (d, e). Otras, sin embargo, requieren algo más de tiempo (a, b, c) para llegar a su estado final.

Una vez que obtengamos una versión del prototipo suficientemente estable y completa se desea realizar un proceso experimental que permita *evaluar a nivel práctico los distintos aspectos del modelo SEM-HP*. Esta evaluación pretende hacerse a dos niveles:

i) Evaluación a nivel de autor.

ii) Evaluación a nivel de usuario.

En el primer nivel **(i)** se desean validar los beneficios derivados del desarrollo en fases del sistema hipermedia y de la capacidad evolutiva del modelo. Prestando especial atención a la usabilidad de la herramienta de autor y a la consistencia de los cambios realizados en el sistema.

En el segundo nivel **(ii)** se pretende comprobar si las técnicas adaptativas empleadas, interfaces y modos de navegación son del agrado de los usuarios y reducen sus problemas de desorientación y sobrecarga cognitiva. En este sentido nos interesa especialmente constatar los beneficios que proporciona la elección personalizada de la estructura de navegación y la disponibilidad de modos alternativos para recorrerla.

En relación con ambos, usuarios (ii) y autor (i), nos interesa evaluar si la técnica de retroalimentación especificada produce información realmente útil para el autor, y si en caso de que éste realice los cambios sugeridos, la redefinición de la estructura hipermedia obedece efectivamente a la imagen que de ésta tienen sus usuarios.

3.2 Educación

Debido a sus características, es inmediata la aplicación de nuestro modelo en el ámbito de los sistemas hipermedia educacionales. Sin embargo, existen algunos aspectos que en este sentido cabría mejorar o ampliar.

En primer lugar, está el problema de ¿cuándo considerar que un ítem ha sido comprendido por el usuario?. Nosotros resolvemos que si el usuario dispone de los prerrequisitos pedagógicos necesarios para asimilar la información del ítem va a ser capaz de hacerlo. No obstante, puede que el usuario seleccione el ítem y después no lea dicha información, la lea sin prestar demasiada atención o simplemente tenga un "día malo" y no se entere bien de su contenido.

En los casos anteriores, y en muchos otros, no debería aplicarse la regla de actualización o al menos no totalmente. Para ello, es necesario establecer mecanismos que permitan comprobar si el usuario efectivamente ha adquirido el conocimiento contenido en el ítem. Estos mecanismos podrían basarse en la realización de *tests o pruebas similares que permitan cuantificar el grado con el que el usuario ha comprendido la información proporcionada*.

Después, caben varias alternativas. La más básica sería ejecutar la regla de actualización sólo si el usuario ha superado un resultado mínimo en la prueba asociada. Otra alternativa, más complicada pero más equitativa, sería ejecutar la regla de actualización ponderando cada actualización incluida en ésta con el resultado de la prueba. En este caso, el conocimiento del usuario sobre un ítem podría decrecer si al releerlo obtiene un resultado inferior al que obtuvo en la lectura anterior.

Con las alternativas propuestas, la adquisición de conocimiento tras la visita de un ítem, depende de la regla de actualización definida por el autor, pero también de la habilidad del usuario para resolver la prueba asociada. Por este motivo, la propiedad de alcanzabilidad sólo podría garantizarse bajo la suposición de que el usuario es capaz de superar la prueba, si se opta por la alternativa básica, o de obtener el máximo resultado, si se considera la segunda alternativa.

En el caso de usar para evaluar el conocimiento de un ítem un *test* o prueba similar de fácil corrección automática, cualquiera de las dos alternativas anteriores podría ser integrada en el modelo sin demasiadas complicaciones. El trabajo más difícil se plantea si se desean incluir pruebas de validación cuya corrección sea subjetiva, como por ejemplo preguntas donde se solicite una reflexión al usuario o problemas que pueden resolverse de modos muy diferentes o incluso tener varias soluciones factibles.

En este caso, se requiere el uso de técnicas de análisis que permitan detectar no sólo si la solución del usuario es correcta o incorrecta, sino también donde ha cometido los errores y a qué motivo pueden deberse éstos. También sería aconsejable la inclusión de técnicas de apoyo a la solución, que asesoren de forma interactiva al usuario mientras que resuelve el problema. Así como la aplicación de técnicas para la solución de problemas basadas en ejemplos.

Asimismo se plantea como trabajo futuro, la *programación de tareas de aprendizaje* que refuercen la instrucción teórica del usuario. Para ello, sería necesario distinguir entre ítems teóricos y prácticos, de modo que un ítem práctico requiere del conocimiento previo de varios ítems teóricos (reglas de conocimiento) y un ítem teórico se ve reforzado por la realización de algunos ítems prácticos (reglas de actualización). La programación podría implementarse siguiendo el mecanismo propuesto para las rutas guiadas, sólo que en éste caso, la ruta hacia la meta deberían contener el mayor número de ítems prácticos posible. También cabría estudiar las relaciones de prerrequisito y refuerzo entre ítems prácticos, y la inclusión de nuevos roles apropiados.

3.3 Colaboración

El modelo SEM-HP es puntualmente "colaborativo" gracias a la técnica de retroalimentación adaptativa, que tiene como objetivo modificar las estructuras de representación para que integren las relaciones semánticas aceptadas mayoritariamente por los usuarios del sistema. La aplicación de esta técnica supone una colaboración inconsciente entre el grupo de usuarios que navegan libremente, ya que el sistema superpone sus estrategias de navegación para extraer las relaciones conceptuales que de forma colectiva tienen en mente.

Evidentemente, la anterior es una forma de colaboración muy particular, ya que no existe ni conciencia, ni comunicación, ni coordinación entre los usuarios que colaboran. En consecuencia, se propone como trabajo futuro la inclusión de los elementos necesarios para conseguir una colaboración real a dos niveles distintos:

i) Colaboración entre autores.

ii) Colaboración entre usuarios finales.

Para hacer efectiva la colaboración a ambos niveles, podría seguirse la metodología AMENITIES (*A MEthodology for aNálisis and desIgn of cooperaTIve systEmS*) [Garrido, 00], basada en modelos de comportamiento y tareas para el análisis, diseño y construcción de sistemas colaborativos.

La colaboración entre autores **(i)** tendría como objetivo la **construcción cooperativa del sistema hipermedia**. Durante esta colaboración, los autores, expertos en un mismo dominio de conocimiento, deberían poder colaborar en la creación y evolución de las estructuras de representación e incluso de las reglas de aprendizaje.

Para ello, habría que definir una serie de reglas o leyes que rijan la distribución del trabajo, la asignación, transición e interrupción de roles, la asignación de tareas a cada rol y la planificación y puesta en marcha de éstas. Además, sería fundamental incorporar herramientas de comunicación que hiciesen más rápido y efectivo el intercambio de información entre los autores. Así como herramientas que permitiesen conocer a un autor qué otros autores están trabajando en ese instante y cuál está siendo su labor.

La colaboración entre usuarios **(ii)** debería permitir, entre otras, las siguientes cosas:

a) Que un usuario pueda observar la estrategia de navegación seguida por otro, con el objetivo de controlar su navegación (rol tutor) o aprender de ésta.

b) Que un usuario pueda saber el grado de conocimiento que otro usuario posee sobre un determinado concepto, con la intención de evaluar su aprendizaje (rol tutor), poder solicitarle ayuda acerca de un concepto que ese usuario conoce bien u ofrecerle la suya en relación a un concepto que desconoce.

c) Que un usuario pueda estar al corriente de los ítems que otro usuario ha leído, con beneficios similares a los expresados en b).

d) Que varios usuarios puedan agruparse y realizar un proceso de *aprendizaje conjunto*, durante el cual se tenga en cuenta el conocimiento individual de cada usuario y el conocimiento del grupo.

Los tres primeros puntos requieren la integración de herramientas de conciencia y monitorización, mientras que para satisfacer el cuarto es necesario, además, enriquecer los formalismos existentes. Por ejemplo, en las reglas de conocimiento habría que indicar para cada prerrequisito si debe ser satisfecho por todos los miembros del grupo, por el usuario actual o por algún usuario del grupo (cualquiera).

También se podría calcular el conocimiento del grupo y evaluar los prerrequisitos de las reglas de conocimiento sobre este conocimiento colectivo, en lugar de tener en cuenta el conocimiento individual de los usuarios que lo forman. En tal caso, habría que definir de qué modo se obtiene el conocimiento del grupo, y modificar las reglas de actualización para determinar si la actualización afecta al usuario que lee el ítem, al grupo o a ambos.

4. LÍNEAS ABIERTAS

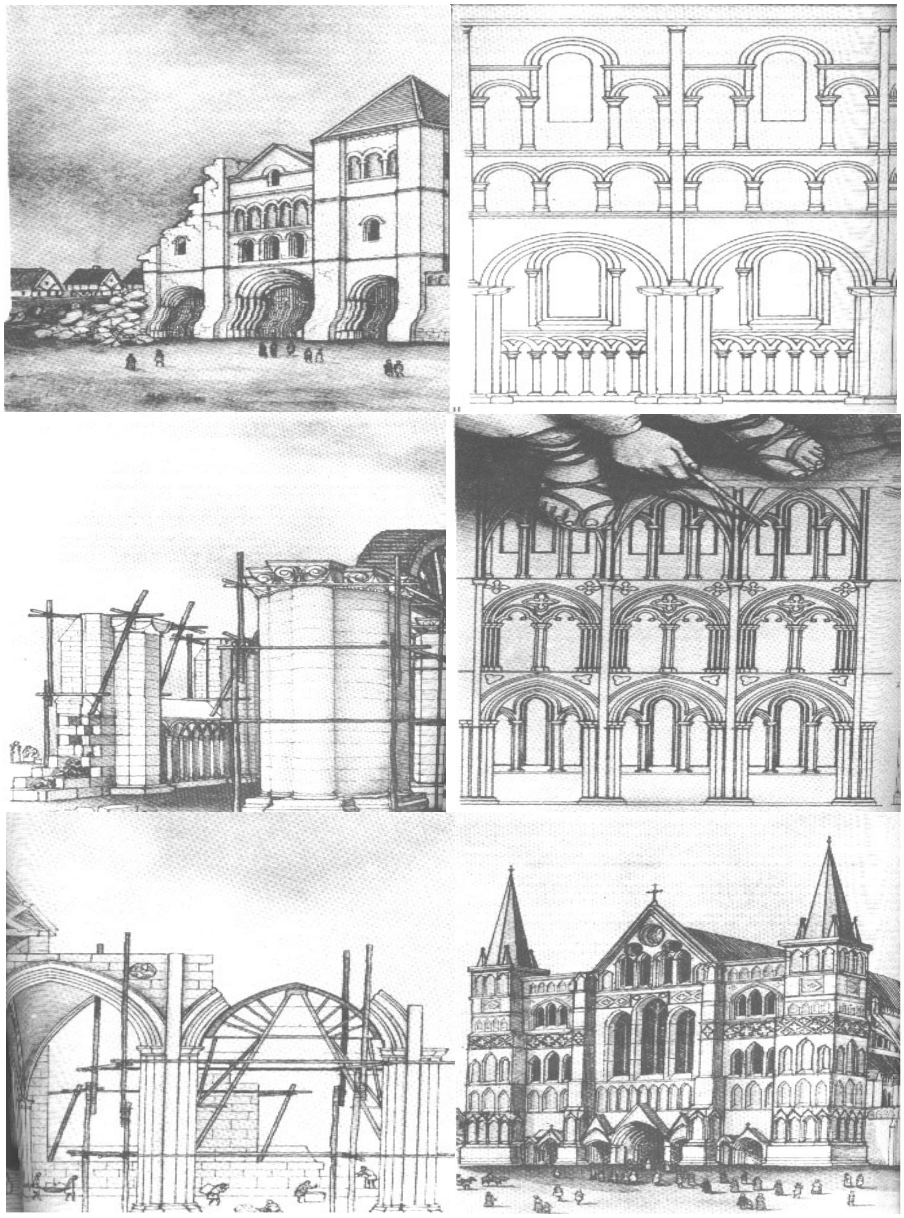
Además del trabajo futuro que se ha proyectado en la sección anterior, recientemente se han abierto una serie de líneas de trabajo que nos están permitiendo aplicar el modelo propuesto en campos muy diversos. De forma muy breve, las líneas son las siguientes:

- Algunas características del modelo están siendo aplicadas para desarrollar el sistema Sc@ut, un sistema hipermeda no anticipativo y adaptativo para la comunicación de niños autistas y personas con necesidades especiales [Rodríguez, 04].

- En colaboración con el departamento de Anatomía Patológica de la Universidad de Granada se está trabajando en un sistema de información hipermedia adaptativo aplicado al estudio de linfomas.

- Parte del trabajo realizado está siendo utilizado en el proyecto titulado “Análisis y Modelado de Sistemas Colaborativos” (TIN 2004-08000-c03-02), el cuál es un subproyecto del proyecto “Sistemas adaptativos y colaborativos con soporte web” (ADACO).

Apéndice



Los Pilares de la Tierra

Tabla de Acrónimos y Abreviaturas

ADAPTS	Adaptive Diagnostics and Personalized Technical Support
AH	Adaptive Hypermedia
AHA	Adaptive Hipermedia Architecture
AHAM	Adaptive Hipermedia Application Model
AHS	Adaptive Hypermedia System
AIES	Adaptive and Intelligent Educational Systems
CAI	Computer Assisted Instruction
CS	Conceptual Structure
DAML	DARPA Agent Markup Language
DTD	Document Type Definition
EC	Estructura Conceptual
GSP	Generic Structure Potencial
HE	Historia estructural
HF	Historia funcional
HTML	Hypertext Markup Language
IA	Interfaz de Acción
IE	Interfaz de Evolución
ITS	Intelligent Tutorial Systems
MEDES	Método de Especificación, Diseño y Evolución del Software
MEMEX	MEMory EXtender
MS	Meta-Sistema
MVC	Modelo Vista Controlador

OIL	Ontolgy Inference Layer
OWL	Web Ontology Language
PUSH	Plan- and User Sensitive Help
RDF	Resource Description Framework
RT	Restricción
SA	Sistema de Aprendizaje
SD	Sistema de decisión
SEM-HP	Modelo Semántico, Sistémico y Evolutivo para el desarrollo de Sistemas Hipermedia.
SGML	Standard Generalized Markup Language
SHA	Sistema Hipermedia Adaptativo
SM	Sistema de Memorización
SN	Sistema de Navegación
SP	Sistema de Presentación
SS	Sistema Software
URI	Uniform Resource Identify
URL	Uniform Resource Locator
WBE	Web-based education
WWW	World Wide Web
XML	eXtensive Markup Language

Tabla de Símbolos

$\mathbf{¢_k}$	Candidato Greedy
Ac	Conjunto de asociaciones conceptuales
$\mathbf{a}_c$	Asociación conceptual
ACe	Acción evolutiva
Af	Conjunto de asociaciones funcionales
$\mathbf{a}_f$	Asociación funcional
C	Conjunto de conceptos
c, $\mathbf{c}_k$	Concepto
$\mathbf{c}_k.\mathbf{i}_j$	Ítem i_j asociado al concepto c_k
Dbls	Conjunto de ítems deseables
$\mathbf{EC}_A$	Estructura conceptual de aprendizaje
$\mathbf{EC}_M$	Estructura conceptual de memorización
$\mathbf{EC}_N$	Estructura conceptual de navegación
$\mathbf{EC}_P$	Estructura conceptual de presentación
$\mathbf{estado}_K$	Estado de conocimiento
$\mathbf{et}_{SEM}$	Etiqueta semántica
g()	Función de costo
$\mathbf{g(i_j, at^i)}$	Bondad del ítem i_j respecto a la característica at^i
$\mathbf{G}_{op\text{-}exp}$	Grado óptimo de un intervalo de experiencia
h()	Función heurística
I	Conjunto de ítems de información
i, $\mathbf{i}_j$	Ítem de información

IExpM, IExpN	Indicadores de experiencia usados para elegir la EC_A
Intervalo$_{Exp}$	Intervalo de experiencia
Ints	Conjunto de ítems interesantes
ISubC	Indicador de subdominio-de-conocimeinto usado para elegir la EC_A
K( )	Función de conocimiento
$\mathbf{K_{max}^{k}(c_m)}$	Conocimiento máximo que es posible alcanzar sobre c_m en EC_P^k
LSubC	Lista de subdominios de conocimiento
M	Meta de conocimiento
$\mathbf{m_j}$	Submeta de conocimiento sobre i_j
$\mathbf{M_k[]}$	Matriz de conocimiento máximo
MT	Matriz de transiciones
$\mathbf{MT_m}$**,** $\mathbf{MT_m}$**-usuarios**	Matriz de transiciones dinámica para la EC_M
$\mathbf{MT_m}$**-autor**	Matriz de transiciones estática para la EC_M
$\mathbf{MT_m}$**-usuarios**$_c$	Matriz de transiciones dinámica de la EC_M durante navegación por conceptos
$\mathbf{MT_p^k}$**,** $\mathbf{MT_p^k}$**-usuarios**	Matriz de transiciones dinámica de una presentación EC_P^k
$\mathbf{MT_p^k}$**-autor**	Matriz de transiciones estática de una presentación EC_P^k
$\mathbf{MT_p^k}$**-usuarios**$_c$	Matriz de transiciones dinámica de una presentación EC_P^k durante navegación por conceptos
MU	Modelo de usuario
$\mathbf{n_k}$	Nodo de un árbol
$\mathbf{P^-}$	Presentaciones menos visitadas
$\mathbf{P^+}$	Presentaciones más visitadas
Rc	Conjunto de relaciones conceptuales
$\mathbf{r_c}$	Relación conceptual

Restricción^{A}	Restricción de accesibilidad
Restricción^{I}	Restricción de idoneidad
Rf	Conjunto de relaciones funcionales
$\mathbf{r}_f$	Relación funcional
Rk	Regla de conocimiento
Rk-autor	Regla de conocimiento con sintaxis de autor
Ro	Regla de orden
RTnb	Restricción de navegabilidad
Ru	Regla de actualización
$\mathbf{Ru(i_j)[s_k]}$	Ejecución de la regla $Ru(i_j)$ sobre el estado de conocimiento s_k
Rw	Regla de peso
S( )	Función subsecuente
$\mathbf{s}_k$	Estado de conocimiento
$\mathbf{Tree}_D$	Árbol de ítems deseables
$\mathbf{Tree}_K$	Árbol de accesibilidad
$\mathbf{visit(i_j)}$	Operador que representa una visita sobre i_j
$\mathbf{w(at^i)}$	Peso asociado al atributo at^i
$\mathbf{w}_j$	Peso del ítem i_j en una regla de peso

Tabla de Definiciones

Acción evolutiva: Operación que permite cambiar los elementos de los distintos sistemas de la arquitectura. Una acción evolutiva debe verificar un conjunto de restricciones para garantizar la consistencia del sistema modificado, y puede implicar la propagación del cambio dentro y fuera de éste.	**6.5**
Árbol de accesibilidad: El árbol de accesibilidad asociado a un ítem i_j, esto es $Tree_K(i_j)$, representa mediante una estructura jerárquica, las restricciones de conocimiento que el usuario debe satisfacer para poder acceder convenientemente al ítem i_j. El árbol integra las restricciones de conocimiento presentes en los antecedentes de accesibilidad de todas las reglas de conocimiento definidas para i_j..	**8.10**
Árbol de ítems deseables: Denominamos $Tree_D$ al árbol que el sistema construye para, dado un conjunto de reglas de conocimiento, un conjunto de ítems interesantes y el estado actual del usuario, identificar el conjunto de ítems deseables. Los nodos del árbol se etiquetan con ítems y las ramas representan restricciones de accesibilidad no satisfechas sobre éstos. Además, la estructura del árbol es tal, que siempre los nodos de primer nivel se etiquetan con los ítems interesantes y los nodos del último nivel contienen los ítems deseables. Dbls = {nodo-hoja($Tree_D$)}.	**19.2**
Concepto: Un concepto es una idea, pensamiento o abstracción que puede ser etiquetado por el autor con el fin de hacer explícito su conocimiento y hacerlo comprensible.	**6.1**
Consistencia Rk ∪ Ru: Diremos que un conjunto de reglas $Rk(EC_A^i) \cup Ru(EC_A^i)$ es consistente si garantiza la alcanzabilidad de todos los ítems incluidos en la EC_A^i, con independencia de la navegación realizada hasta el momento y el estado de conocimiento inicial del usuario.	**10.2**
Dominio conceptual: Un dominio conceptual es un conjunto de conceptos con los que se pueden identificar los distintos ítems de información ofrecidos en un sistema hipermedia, y el conjunto de asociaciones semánticas que se pueden establecer entre ellos.	**6.3**
Dominio de conocimiento: El dominio de conocimiento del sistema hipermedia se representa en la estructura conceptual de memorización, y constituye la unión de un dominio conceptual y un dominio de información definido sobre éste.	**6.9**
Dominio de información: Un dominio de información es el conjunto de trozos de información identificados con conceptos que pertenecen a un dominio conceptual concreto, incluyendo la asociación que mantienen con el concepto	**6.4**
Esquema del MU: El esquema del modelo de usuario es el diseño global de éste, es decir la forma en que se representan y organizan los distintos elementos que lo componen. En definitiva la estructura del modelo. Existe un esquema del modelo de usuario único para cada sistema hipermedia. Los cambios en el esquema están	**12.8**

motivados por las modificaciones que el autor realiza en la estructura conceptual de memorización de dicho sistema hipermedia.	
Estado de conocimiento: Un estado de conocimiento es una composición de múltiples conocimientos. El estado de conocimiento al que aquí haremos referencia se limita únicamente al dominio del sistema hipermedia. Por lo tanto, está formado por el conocimiento que éste posee acerca de cada trozo de información existente en el sistema hipermedia. Su estado de conocimiento varía cada vez que el usuario modifica (aumenta o disminuye) algún conocimiento incluido en dicho estado. $estadoK = \bigcup_{i_j \in I(EC_M)} \langle i_j, K(i_j) \rangle$ donde $I(EC_M)$ es el dominio de información de EC_M	**8.2, (1)**
Estructura conceptual de aprendizaje: Una estructura conceptual de aprendizaje queda perfectamente definida a partir de una tupla $EC_A = (EC_M, Rw, MU, EC_N^j, Ru, Rk)$, de forma extendida $EC_A = (EC_M, Rw, MU, EC_P^i, RTnb^j, Ro^j, Ru, Rk)$, donde se establece un conjunto de reglas de peso y un modelo de usuario para la EC_M, y se define un conjunto de reglas de orden, de navegabilidad, de conocimiento y de actualización para una de sus presentaciones, EC_P^i.	**6.15**
Estructura conceptual de memorización: La estructura conceptual de memorización es una tupla $EC_M = (C, I, Rc, Rf, Ac, Af)$, que representa íntegramente a través de sus elementos el dominio conceptual y de información del sistema hipermedia.	**6.8**
Estructura conceptual de navegación: Una estructura conceptual de navegación se define con una tupla $EC_N = (EC_P^i, RTnb, Ro)$, donde el primer elemento representa una estructura conceptual de presentación, y los dos últimos un conjunto de restricciones de navegabilidad y reglas de orden que limitan parcialmente la forma en que ésta va a poder recorrerse.	**6.13**
Estructura conceptual de presentación: Una estructura conceptual de presentación se define mediante una tupla $EC_P = (C^p, I^p, Rc^p, Rf^p, Ac^p, Af^p)$ que contiene un subconjunto de los conceptos (C), ítems (I), relaciones conceptuales (Rc), relaciones funcionales (Rf), asociaciones conceptuales (Ac) y asociaciones funcionales (Af) presentes en la estructura conceptual de memorización.	**6.11**
Estructura conceptual: Una estructura conceptual se define formalmente mediante una tupla EC = (C, I, Rc, Rf, Ac, Af), en la que C es un conjunto de conceptos, I es un conjunto de ítems de información, Rc es un conjunto de relaciones conceptuales, Rf es un conjunto de relaciones funcionales, Ac es un conjunto de asociaciones conceptuales y Af es un conjunto de asociaciones funcionales.	**6.7**
Estructura de composición: La forma en que se reúnen en un único "documento" el conjunto de ítems que integran el dominio de información asociado directamente a un concepto constituye la estructura de composición del resumen generado para éste.	**16.2**
Etiquetado de EC_A: El etiquetado de una estructura conceptual de aprendizaje, EC_A, es un conjunto de *n* parejas ($atributo^i$, $etiqueta^i$), donde, desde i = 1. .n, la $etiqueta^i$ refleja el valor que dicha estructura tiene para la característica $atributo^i$. Esta	**14.1**

información se usa para elegir la EC_A que se proporciona a un usuario de acuerdo a su perfil.	
Experiencia de navegación: La experiencia de navegación representa la práctica del usuario en el uso de sistemas hipermedia.	**12.3**
Experiencia en la materia: La experiencia en la materia trata de identificar con qué grado conoce o cómo está de familiarizado el usuario con el conjunto de conceptos sobre los que versa el material ofrecido en el sistema hipermedia, es decir, cuál es su experiencia en el tema desarrollado en dicho sistema.	**12.2**
Función de conocimiento: Para referenciar el grado de conocimiento acerca de un elemento x, definimos una función de conocimiento, notada como K(x), que representa en valor numérico el grado de conocimiento acerca del ítem o concepto identificado por x.	**8.5**
Grado de conocimiento: El conocimiento del usuario acerca de un ítem o concepto se mide y expresa utilizando un grado de conocimiento. Este grado de conocimiento se representa mediante una etiqueta semántica, a la que de forma abreviada notaremos et_{SEM}. Dicha etiqueta refleja el nivel de conocimiento del usuario, es decir, cuánto conoce acerca de un determinado elemento, abstracto o concreto. Puede tomar los valores "nulo"(0), "bajo"(1), "medio"(2), "alto"(3) y "total"(4).	**8.3, 8.4**
Instancia del MU: Una instancia del modelo de usuario es la colección de información almacenada en el modelo para un usuario concreto. En un sistema hipermedia existen tantas instancias del modelo de usuario como usuarios diferentes se hayan registrado en el mismo. Una instancia del modelo de usuario es una copia del esquema donde en cada entrada se rellenan los datos de ese usuario particular. Por lo tanto, una instancia cambia con tanta frecuencia como el usuario al que representa.	**12.9**
Ítem alcanzable: Un ítem i_k incluido en una estructura de aprendizaje EC_A^i, es alcanzable si a través de la navegación de ésta, cualquier usuario puede conseguir un estado de conocimiento para el cuál se satisfacen las restricciones de accesibilidad de al menos una de las reglas de conocimiento, $Rk(i_k)$, asociadas a i_k en $Rk(EC_A^i)$.	**10.1**
Ítem deseable: Un ítem deseable es aquél cuya visita es provechosa desde la perspectiva de visitar los ítems interesantes o alcanzar la meta de conocimiento deseada.	**19.1**
Ítem interesante / Concepto interesante: Un ítem o concepto interesante es simplemente aquél por cuyo conocimiento el usuario tiene una especial predilección. Siempre y cuando sea marcado como tal en su modelo de usuario. El conjunto de todos los ítems y conceptos interesantes es denominado *Ints*.	**12.4**
Ítem meta / Concepto meta: Un ítem o concepto meta es todo aquél para el que existe una submeta de conocimiento en M. Es decir, i_j es un ítem meta si $\exists m_j \in M$, donde $m_j = (i_j, et_{SEM}^j)$ y c_j es un concepto meta si $\exists m_j \in M$, donde $m_j = (c_j, et_{SEM}^j)$.	**12.7**

Ítem: Un ítem es cualquier trozo de información identificable en un sistema hipermedia.	**6.2**
Matriz de transiciones de autor para EC_M: La matriz de transiciones MT_m-autor es una matriz estática que recoge la existencia o ausencia de relaciones semánticas entre cada dos conceptos de la EC_M definida por el autor. El valor de una celda MT_m-autor[c_i, c_j] puede ser 1 o 0 según exista o no una relación desde c_i hasta c_j. Este valor sólo es reconsiderado cuando el autor utiliza acciones evolutivas en el Sistema de Memorización que puedan afectar al dominio conceptual de la EC_M.	**21.5**
Matriz de transiciones de autor para EC_P^k: El valor de cada una de las $n \times n$ celdas de la matriz estática MT_p^k-autor indica la existencia o ausencia de una determinada relación conceptual en la presentación EC_P^k definida por el autor. Concretamente, la celda MT_p^k-autor[c_i, c_j] es igual a 1 si se han incluido en la EC_p^k una o más relaciones conceptuales que tienen como origen el concepto c_i y como destino el concepto c_j, en otro caso es 0.	**21.6**
Matriz de transiciones de EC_M: La matriz de transiciones de una estructura conceptual de memorización EC_M se nota como MT_m y contiene las relaciones definidas entre cada dos conceptos por el conjunto de usuarios que recorren cualquiera de sus presentaciones en modo tradicional. Concretamente, para calcular la matriz de transiciones MT_m, el sistema combina las matrices de transiciones, MT_p^1, MT_p^2,..., MT_p^r, de las *r* presentaciones EC_P^1, EC_P^2,..., EC_P^r definidas a partir de EC_M.	**21.4**
Matriz de transiciones de EC_P^k: La matriz de transiciones asociada a una presentación tiene una fila y una columna por cada concepto incluido en ésta. Así, siendo $C(EC_P^k)=\{c_1, c_2, ..., c_n\}$ el conjunto de conceptos mostrados en la presentación EC_P^k, la matriz MT_p^k asociada es una matriz cuadrada de orden *n*, que contiene por tanto $n \times n$ elementos.	**21.3**
Matriz de transiciones: Denominamos matriz de transiciones (MT) a la estructura de datos utilizada para identificar las asociaciones conceptuales que el grupo de usuarios tiene en mente durante su navegación por una red semántica.	**21.2**
Meta de conocimiento: Una meta de conocimiento, denominada M, es un conjunto de parejas formadas por un ítem o concepto y un grado de conocimiento. Esto es, $M = \{m_i, m_j,..., m_k\}$. Cada una de las parejas incluida en la meta, $m_j \in M$, es una submeta que establece el grado de conocimiento mínimo (expresado mediante una de las etiquetas semánticas disponibles) que el usuario desea lograr sobre el ítem o concepto involucrado.	**12.5**
Modelo de usuario: El modelo de usuario, al que de forma abreviada notamos MU, es precisamente, la representación interna que el sistema mantiene del usuario para posteriormente ser capaz de adaptar su estructura y comportamiento a las características y necesidades del mismo.	**12.1**
Navegación libre: En la navegación libre, el usuario puede seleccionar cualquiera de las unidades de navegación que aparecen en la estructura actual, es decir, no tiene que satisfacer ningún prerrequisito previo para poder acceder a la información	**15.1**

ofrecida.	
Navegación por conceptos: La navegación por conceptos es un modo de navegación libre, donde la estructura proporcionada para navegar muestra únicamente conceptos y relaciones conceptuales. La unidad de navegación es el concepto, de modo que al seleccionar un concepto, el usuario obtiene un resumen compuesto con varios ítems de información asociados a éste.	**16.1**
Navegación por conocimiento: La navegación por conocimiento es un modo de navegación restringido, que establece la accesibilidad e idoneidad de un ítem en función del estado de conocimiento del usuario y los requisitos pedagógicos impuestos por el autor para su visita. La unidad de navegación es el ítem y la red semántica proporcionada coincide con la EC_P^k incluida en la EC_A^j elegida para el usuario.	**18.1**
Navegación por relación conceptual: La navegación por relación conceptual es un modo de navegación restringido, que define una forma de recorrer la estructura de navegación coherente con las relaciones semánticas existentes entre sus conceptos. La unidad de navegación es el ítem y la red semántica proporcionada para navegar coincide con la EC_P^k incluida en la EC_A^j elegida para el usuario.	**17.1**
Navegación restringida: En la navegación restringida, el usuario tiene prohibido el acceso a determinados ítems si no cumple una serie de condiciones necesarias. Éstas son establecidas por el autor y dependen de cada modo y estructura de aprendizaje concreta.	**15.2**
Navegación tradicional: La navegación tradicional es un modo de navegación libre, donde la red semántica proporcionada para navegar coincide con la estructura conceptual de presentación EC_P^i incluida en la estructura de aprendizaje elegida para el usuario, EC_A^j. Permite acceder al contenido de cualquier ítem sin más que seleccionarlo en la red. No tiene pues restricciones de acceso para acceder a la información y la unidad de navegación es el ítem.	**15.3**
Perfil de usuario: De acuerdo a lo expuesto, consideramos que el perfil de un usuario está determinado por su grado de experiencia, tanto en la materia del actual sistema hipermedia como en la navegación de sistemas web en general. Además, de por el subdominio de conocimiento en el que dicho usuario está más interesado.	**14.2**
Propagación externa del cambio: El mecanismo de propagación externa del cambio es el que se ocupa de garantizar que todos los sistemas co-evolucionen en una forma consistente después de que el autor realice una modificación en cualquiera de ellos. De esta forma además de la consistencia individual de cada sistema se garantiza la consistencia global del conjunto, asegurando pues, la integridad del sistema hipermedia en todo momento.	**13.1**
Regla de actualización 1: La regla de actualización asociada al ítem i_j determina, después de una visita a i_j, cuál es el incremento de conocimiento del usuario respecto a i_j, y posiblemente a otros ítems relacionados. **Regla de actualización 2**: Existe un conjunto de reglas de actualización para cada estructura conceptual de aprendizaje EC_A^i, al que notamos de forma abreviada	**9.1, 9.2**

Ru(EC_A^i). La cardinalidad del conjunto Ru(EC_A^i) coincide con el número de ítems mostrados en la estructura conceptual de presentación sobre la que se define la EC_A^i. Concretamente existe una regla de actualización para cada ítem i_j incluido en dicha presentación, a la cuál notamos Ru(i_j) de forma unívoca dentro de la EC_A^i.	
Regla de conocimiento: Una regla de conocimiento define las restricciones de conocimiento (prerrequisitos pedagógicos) que debe cumplir el usuario para poder visitar un determinado ítem y para que dicha visita sea aconsejable.	**8.1**
Regla de peso: La regla de peso Rw(c_k) ligada a un concepto c_k, calcula el grado de conocimiento del usuario sobre dicho concepto a partir del grado de conocimiento que éste posee sobre cada uno de los ítems asociados funcionalmente al concepto en la EC_M. Obviamente, el contenido de dos ítems asociados a un mismo concepto no tiene porqué ser de igual relevancia para el conocimiento de éste. Por esta razón, en la regla de peso se pondera la importancia de cada ítem implicado en el cálculo del conocimiento sobre el concepto.	**11.1**
Restricción de accesibilidad: Una restricción de accesibilidad, a la que llamamos $Restricción^A$, establece cuál es el grado de conocimiento mínimo que debe poseer el usuario sobre un ítem para que el acceso a otro determinado ítem esté *permitido*. El predicado para especificar una restricción de este tipo es de la forma: K(i_j) ≥ val, donde la variable i_j será instanciada con el identificador de un ítem de información y la variable *val* con una etiqueta semántica.	**8.8**
Restricción de idoneidad: Una restricción de idoneidad, notada como $Restricción^I$, determina cuál es el grado de conocimiento máximo o exacto que debe poseer el usuario sobre un ítem para que la visita a otro determinado ítem sea *aconsejable*. Existen dos predicados para expresar una restricción de este tipo: K(i_j) ≤ val y K(i_j) = val. En ambos casos, i_j se instancia con un ítem y *val* con una etiqueta de conocimiento.	**8.9**
Retroalimentación adaptativa: Nombramos retroalimentación adaptativa a la técnica empleada por el Sistema de Aprendizaje para: 1) analizar e integrar el comportamiento navegacional de los usuarios que recorren el sistema hipermedia en modo libre, 2) cotejar la estructuras trazadas por éstos con las estructuras definidas previamente por el autor y 3) sugerir las modificaciones necesarias para hacer converger ambas estructuras.	**21.1**
Rk-autor 1: Dentro de una estructura conceptual de aprendizaje concreta notaremos Rk-autor(i_j) a una regla de conocimiento asociada al ítem i_j. Para diferenciar las distintas reglas de conocimiento definidas por el autor para un mismo ítem las numeramos con un superíndice, como sigue: Rk-autor1(i_j), Rk-autor2(i_j), Rk-autor3(i_j),...	**8.6**
Rk-autor 2: Notamos Rk-autor(EC_A^i) al conjunto de reglas de conocimiento definidas por el autor en una estructura conceptual de presentación específica, EC_A^i. Este conjunto pone de manifiesto las relaciones de aprendizaje existentes entre los distintos ítems de su dominio de información. Por lo tanto, las reglas del conjunto deben hacer referencia a ítems mostrados en la estructura conceptual de presentación	**8.7**

sobre la que se define la EC_A^i.	
Ruta guiada: Una ruta guiada es una secuencia de ítems, esto es, una lista de ítems y un orden estricto para visitarlos. La ruta es generada por el sistema a partir de: el estado de conocimiento actual del usuario, la meta especificada por éste y sus preferencias. Y siempre se cumple que, después de recorrer la ruta, el estado de conocimiento del usuario satisface completamente la meta.	**20.1**
Sistema de Aprendizaje: El Sistema de Aprendizaje es uno de los cuatro sistemas del modelo SEM-HP. Se encarga de modelar al usuario y adaptar la estructura y el funcionamiento del sistema hipermedia a sus características personales.	**6.14**
Sistema de Memorización: El Sistema de Memorización es uno de los cuatro sistemas del modelo SEM-HP. Establece la materia prima con la que se ha de construir el sistema hipermedia.	**6.6**
Sistema de Navegación: El Sistema de Navegación es uno de los cuatro sistemas del modelo SEM-HP. Permite restringir la navegación de una presentación en un orden coherente con las relaciones conceptuales incluidas en ésta.	**6.12**
Sistema de Presentación: El Sistema de Presentación es uno de los cuatro sistemas del modelo SEM-HP. Permite seleccionar una parcela de conocimiento, con el fin de presentar una estructura de navegación de tamaño reducido.	**6.10**
Starting-point: Dada una presentación EC_P^k y un conjunto de reglas de navegabilidad $RTnb^i$, denominamos punto de partida o *starting-point* a un concepto que no es destino de ningún enlace de navegación y que por lo tanto siempre está disponible durante la navegación por relación conceptual, independientemente de cuál sea el último concepto visitado.	**17.2**
Submeta de conocimiento: Una submeta de conocimiento m_j se define sobre un elemento e_j y tiene la forma: (e_j, et_{SEM}^j). La submeta m_j es satisfecha cuando el grado de conocimiento que el usuario posee sobre el elemento e_j es mayor o igual que el indicado en et_{SEM}^j. Esto es, m_j se satisface sólo si $K(e_j) \geq$ valor-numérico(et_{SEM}^j). El elemento e_j puede ser un ítem, i_j, o un concepto, c_j.	**12.6**

Bibliografía

[Alatalo, 01] Alatalo, T.; Peräaho, J. "A Modelling Method for Designing Adaptive Hypermedia". 3rd Workshop on Adaptive Hypertext and Hypermedia. UM'2001. Sonthofen, Germany. July 13-17, 2001.

[Alberts, 93] Alberts, L.K. "YMIR: an Ontology for Engineering Design". University of Twente, Ph Thesis, 1993.

[Alder, 01] Alder, Gaudenz. "The JGraph Swing Component". http://www.jgraph.com © 2001-2004.

[Amano, 00] Amano, M.; Watanabe, T."An Approach for constructing Component-base Software Systems with Dynamic Adaptability using LEAD++". Principles of Software Evolution. IPSE 2000. IEEE Computer Society. Pp:118-127. 2000.

[Aoyama, 00] Aoyama, M. "Evolutionary Patterns of Desing and Design Patterns". Principles of Software Evolution. IPSE 2000. IEEE Computer Society. Pp:110-117. 2000.

[Ardissono, 01] Ardissono, L.; Console, L.; Torre, I. "Exploiting User Models for Personalizing News Presentations". 7th International Conference on User Modeling. Banff, Canada, June 20-24, 2001.

[Ardissono, 01b] Ardissono, L.; Goy, A.; Petrone, G.; Segnan, M.; Torasso, P. "Tailoring the Recommendation of Tourist Information to Heterogeneous User Groups." 3rd Workshop on Adaptive Hypertext and Hypermedia. Hypertext'01. Aarhus, Denmark. August 14-18, 2001.

[Aroyo, 04] Aroyo, L.; De Bra, P.; Chepegin, V. "Semantic Web-based Adaptive Hypermedia". Proceedings of WWW'04. Workshop on Applications in the Semantic Web. 2004.

[Bailey, 01] Bailey, C.; El-Beltagy, S.; Hall, W. "Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia". 3rd Workshop on Adaptive Hypertext and Hypermedia. Hypertext'01. Aarhus, Denmark, August 14-18, 2001.

[Balasubramanian, 93] Balasubramanian, V. "Hypermedia Issues and Applications: A State of the Art Review". Independent Research Report as part of Ph.D. Program, Graduate School of Management, Rutgers University. December, 1993.

[Ballesteros, 03] Ballesteros-Olmos, Romina. Un pensamiento científico en web. http://soko.com.ar

[Banerjee, 87] Banerjee, J.; Kim, V.; Kim, H.K.; Korth, H.F. "Semantics and Implementation of Schema Evolution in Object-Oriented Databases". In Proceedings of ACM-SIGMOD International Conference on Management of Data. San Francisco CA. May, 1987.

[Belady, 76] Belady, L.A.; Lehman, M. M. "A Model of Large Program Development". IBM Syst. J. Volumen (15) 3, Pp: 225-252. 1976.

[Berners-Lee, 00] Berners-Lee, T. "Semantic Web - XML 2000". Keynote Speech. http://www.w3.org/2000/Talks/1206-xml2k-tbl

[Berners-Lee, 01] Berners-Lee, T.; Hendler, J.; Lassila, O. "The Semantic Web". Scientific American, Volumen 284, nº. 5. Pp: 34-43. May, 2001.

[Bertalanffy, 68] Bertalanffy, L.Von. "Gerneral System Theory". New York. George Brazilller, Inc, 1968. (Trad. Fr. Théorie générale des sytèmes. Paris. Dunod. 1973).

[Bertalanffy, 75] Bertalanffy, L.Von. "Perspective on General System Theory". New York. George Brazilller, Inc, 1975. (Trad. Es. Perspectivas en la Teoría General de Sistemas. Madrid. Alianza Universidad. 1979).

[Berzins, 93] Berzins, V.; Luqi, T.; Yehudai, A. "Using Transformations in Specification-Based Prototyping". IEEE Trans.S.E. Volumen 19.5. Pp:436-452. 1993.

[Bitzer, 71] Bitzer, D. L.; Johnson, R. L. "Plato. a computer-based system used in the engineering of education," Proceedings of the IEEE. Volume 59. No. 6. Pp: 960–968. 1971.

[Boehm, 88] B.W. Boehm. "A Spiral Model of Software Development and Enhancement". In IEEE Computer. Volumen 21(5). Pp. 61-72. May, 1988.

[Bollen, 96] Bollen, J.; Heylighen, F. "Algorithms for the self-organisation of distributed, multi-user networks. Possible application to the future World Wide Web". Cybernetics and Systems '96. R. Trappl (ed.). (Austrian Society for Cybernetics), Pp: 911-916.

[Bollen, 98] Bollen, J.; Heylighen, F. "A System to Restructure Hypertext Networks into Valid User Models". The New Review of Hypermedia and Multimedia. Volume 4. Pp:189-213. 1998.

[Bollen, 00] Bollen, J. "Group User Models for Personalized Hyperlink Recommendations". International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2000). In LNCS, Volume 1892. Pp: 39-50. Springer Verlag. Trento, August 2000.

[Brassard, 96] Brassard, G.; Bratley, P. "Fundamentals of Algorithmics". Prentice Hall, 1996.

[Brusilovsky, 96] Brusilovsky, P. "Methods and Techniques of Adaptive Hypermedia". User Modeling and User-Adapted Interaction, 6. Pp: 87-129. 1996. (Reprinted in Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers. Pp:1-43. 1998).

[Brusilovsky, ELM] Brusilovsky, P.; Weber, G. "Episodic Learner Model – The HomePage ELM". http://www.psychologie.uni-trier.de:8000/projects/ELM/elm.html

[Brusilovsky, 99] Brusilovsky, P.; Cooper, D. "ADAPTS: Adaptive Hypermedia for a Web-based Performance Support System". 2nd Workshop on Adaptive Systems and User Modeling on the WWW. Canada, 1999.

[Brusilovsky, 99b] Brusilovsky, P. "Adaptive and Intelligent Technologies for Web-based Education". In C.Rollinger and C.Peylo(eds.). Special Issue on Intelligent Systems and Teleteaching, Künstliche Intelligenz. Volume 4. Pp: 19-25. 1999.

[Brusilovsky, 00] Brusilovsky, P. "Adaptive Hypermedia: From Intelligent Tutoring Systems to Web-Based Education". Intelligent Tutoring Systems 2000. Pp: 1-7. ITS'00. Montrél, Canada. 2000.

[Brusilovsky, 02] Brusilovsky, P.; Cooper, D. "Domain, Task, and User Models for an Adaptive Hypermedia Performance Support System". Proceedings of the 7th international conference on Intelligent User Interfaces. Pp: 23- 30. ISBN:1-58113-382-0. San Francisco, California, USA. 2002.

[Bush, 45] Bush V. "As We May Think". The Atlantic Monthy, 176. Pp: 101-108. July, 1945.

[Bylander, 98] Bylander, T.; Chandrasekaran, B. "Generic Tasks in Knowledge-Based Reasoning: The Right Level of Abstraction for Knowledge Acquisition". In B. R. Gaines and J.H. Boose Eds., Knowledge Acquision for Knowledge Based Systems. Academic Press, London, 1988.

[Cañas, 00] Cañas, A.; Ford, K.; Coffey, J.; Reichherzer, T.; Suri, N.; Carff, R.; Shamma, D.; Hill, G.; Breedy, M. "Herramientas para Construir y Compartir Modelos de Conocimiento Basados en Mapas Conceptuales". Revista de Informática Educativa, Vol. 13, No. 2 (2000), pp. 145-158.

[Carbonell, 70] Carbonell, J.R. "AI in CAI: An Artificial Intelligence Approach to Computer-assisted Instruction". IEEE Transactions on Man-Machine Systems. Volume 11. Pp: 190-202. 1970.

[Carretero, 94] Carretero, M. "Constructivismo y Educación". Editorial Aique. Buenos Aires, 1994.

[Casais, 90] Casais, E. "Managing Class Evolution in Object-Oriented Systems". In Object Management. Pp: 133-195. Tsichritzis, D. GENEVE. Centre Universitaire d'Informatique. 1990.

[Castell, 02] Castell, P. "Aplicación de Técnicas de la Web Semántica". II Workshop de Investigación sobre nuevos paradigmas de interacción en entornos colaborativos. COLINE'02. Granada, 11 y 12 de Noviembre de 2002.

[Castell, 02b] Castells, P.; Macías, J.A. "Un Sistema de Presentación Dinámica en Entornos Web para Representaciones Personalizadas del Conocimiento". Revista Iberoamericana de Inteligencia Artificial. Nº. 16. Pp: 25-34. AEPIA, 2002.

[Coltell, 99] Coltell, O.; Ortells, S.; Guillén, M. "Tutorial Inteligente para el Aprendizaje de Epidemiología Genética y Molecular". Comunicación en

INFORSALUD '99. III Congreso Nacional de Informática de la Salud. Madrid, 2 al 4 de Marzo de 1999.

[Conklin, 87] Conklin, J. "Hypertext: An Introduction and Survey". IEEE Comuter, Volumen 20 (9). Pp:17-41. 1987.

[Cornen, 97] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. "Introduction to Algorithms". McGraw Hill / MIT Press, 1997.

[DAEDALUS, 2004] © 2004 DAEDALUS - Data, Decisions and Language, S. A. http://www.daedalus.es

[Dancy, 93] Dancy, Jonathan. "Introducción a la epistemología contemporánea". Edición Tecnos. Madrid, 1993.

[DaSilva, 97] Da Silva, P.; Van Durm, R.; Duval, E.; Olivié, H. "A Simple Model for Adaptive Courseware Navigation". Proceedings of INFWET97. Eindhoven. November, 1997.

[DaSilva, 98] Da Silva, P.; Van Durm, R.; Duval, E.; Olivié, H. "Concepts and Documents for Adaptive Educational Hypermedia: a Model and a Prototype". Second Workshop on Adaptive Hypertext and Hypermedia. Ninth ACM Conference on Hypertext and Hypermedia. Pp: 35-43. Pittsburgh, USA. Junio 20-24, 1998.

[DeBra, 98] De Bra, P.; Houben, G.J.; Wu, H. "AHAM: A Reference Model to Support Adaptive Hypermedia Authoring". InfWet 98 Conference, 1998.

[DeBra, 98b] De Bra, P.; Calvi, L. "AHA: a Generic Adaptive Hypermedia System". Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia. Pp: 5-12, Pittsburgh, 1998.

[DeBra, 98c] De Bra, P. "Adaptive Hypermedia on the Web: Methods, Technology and Applications". Proceedings of the AACE WebNet 98 Conference. Pp: 220-225. Orlando, Florida. 1998.

[DeBra, 99] De Bra, P.; Brusilovsky, P. "Adaptive Hypermedia: From Systems to Framework". ACM Computing Surveys. Volume 31(4). Art. 12. December 1999.

[DeBra, 00] De Bra, P.; Aerts, A.; Houben, G.J.; Wu, H. "Making General-Purpose Adaptive Hypermedia Work". Proceedings of the WebNet Conference. Pp: 117-123. 2000.

[DeBra, 02] De Bra, P.: Aerts, A.; Smits, D.; Stash, N. "AHA! meets AHAM". Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems. Berlin, Springer. Pp: 381-384. May, 2002.

[Descartes, 1596-1650] Revista Investigación y Ciencia. Edición especial: "Grandes matemáticos". Prensa científica, SA.

[Echeverría, 1988] Echeverría, R. "El Buho de Minerva**:** Introducción a la Filosofía Moderna". PIIE. Santiago. 1988.

[Engelbart, 63] Engelbart, D. “A Conceptual Framework for the Augmentation of Man’s Intellect”. Vistas in Information Handling, Spartan Books, 1963.

[Espinoza, 96] Espinoza, F.; Höök, K. “A WWW Interface to an Adaptive Hypermedia System”. Conference on Practical Application of Agent Methodology. London, 1996.

[Fairley, 87] Fairley, R. “Ingeniería del Software”. McGraw-Hill, 1987.

[Felici, 03] Felici, M. "Taxonomy of Evolution and Dependability", USE 2003. Second International Workshop on Unanticipated Software Evolution. Short paper. ETAPS’03. Warsaw, Poland. April 5 - 13, 2003.

[García, 95] García Cabrera, L.; Rivero Cejudo, M.L. “Reflexiones sobre el uso del Computador en la Educación”. Novática. Pp:48-51. 1995.

[García, 96] García Cabrera, L.; Rivero Cejudo, M.L. “Educación e Hipermedia”. II Jornadas de Informática. Pp:349-358. Granada. Julio, 1996.

[García, 97] García Cabrera, L.; Parets Llorca, J. “La Integración de Hipermedia en Sistemas de Información: Una Cuestión de Semántica”. III Jornadas de Informática. Pp: 245-256. Cádiz. Julio, 1997.

[García, 00] García Cabrera, L.; Parets Lorca, J. “A Cognitive Model for Adaptive Hypermedia Systems”. 1st International Conference on WISE. Workshop on World Wide Web Semantics. Pp: 29-33. Hong-Kong, China. June, 2000.

[García, 01] García Cabrera, L.; Rodríguez Fórtiz, M.; Parets Llorca, J. “Formal Foundations for the Evolution of Hypermedia Systems”. 5th European Conference on Software Maintenance and Reengineering. Workshop on FFSE. IEEE Press. Pp: 5-12. Lisbon. March, 2001.

[García, 01b] García Cabrera, L.; Rodríguez Fórtiz, M.; Parets Llorca, J. “Toward a Formalisation of Evolutionary Hypermedia Systems based on Systems Theory”. Eight International Conference on Computer Aided Systems Theory. EUROCAST’01. LNCS, 2178. Springer-Verlag. 2001.

[García, 01c] García Cabrera, L. “SEM-HP: Un Modelo Sistémico, Evolutivo y Semántico para el desarrollo de Sistemas Hipermedia”. Tesis Doctoral. Noviembre, 2001.

[García, 02] García Cabrera, L.; Rodríguez Fórtiz, M.; Parets Llorca, J. “Evolving Hypermedia Systems: a Layered Software Architecture”. Journal of Software Maintenance and Evolution: Research and Practice. John Wiley & Sons, Ltd. 14(5), Pp: 389-405.

[Garrido, 00] Garrido, J.L.; Gea, M.; Gutiérrez, F.; Padilla, N. “Designing Cooperative Systems for Human Collaboration”. In Dieng, R., Giboin, A., Karsenty, L., De Michelis, G (eds.): Designing Cooperative Systems – The Use of Theories and Models. IOS Press-Ohmsha 2000.

[Gettier, 63] Gettier, E. "Is Justified True Belief Knowledge?", Analysis 23 (1963): 121-23.

[Goldfarb, 90] Goldfarb, C.F. "The SGML Handbook". Edited and with a foreword by Yuri Rubinsky. Oxford University Press. Oxford, 1990.

[Goldfarb, 00] Goldfarb, C.F. "The XML Handbook". 2nd Edition. Upper Saddle River. Prentice Hall. New Jersey, 2000.

[Gruber, 95] Gruber, T. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". International Journal of Human and Computer Studies, 43 (5/6), Pp: 907-928. 1995.

[Gulbransen, 98] Gulbransen, D. "HTML dinámico: edición especial" Prentice may Iberia. Madrid[etc.], 1998.

[Halasz, 90] Halasz, F.; Schwartz, M. "The Dexter Reference Model". Proceedings of the NIST Hypertext Standardization Workshop. Pp:95-133. 1990.

[Halliday, 85] Halliday, M.A.K. "Introduction to Functional Grammar". Edward Arnold, 1985.

[Halliday, 85b] Halliday, M.A.K.; Ruquaiya, H. "Language, Context and Text: Aspects of Language in a Social-semiotic Perspective". Deaking University Press, 1985.

[Harold, 98] Harold, E.R. "XML: Extensible Markup Language". Inc. IDG Books Worldwide, cop, 1998.

[Heckel, 01] Heckel, R.; Engels, G. "Transformation as a Meta Language for Dynamic Modelling and Model Evolution". Formal Foundations for the Evolution of Hypermedia Systems. 5th European Conference on Software Maintenance and Reengineering, Workshop on FFSE. IEEE Press. Pp: 42-47. Lisbon, Portugal. March 2001.

[Henze, 99] Henze, N.; Nejdl, W. "Bayesian Modeling for Adaptive Hypermedia Systems". ABIS99, 7.GI - Workshop Adaptivität und Benutzermodellierung in Interaktiven Softwaresystemen 29./30.9. Otto-von-Guericke-Universität Magdeburg. 1999.

[Henze, 01] Henze, N. "Open Adaptive Hypermedya: An approach to adaptive information presentation on the Web". 1st International Conference on Universal Access in Human-Computer Interaccion (UAHCI 2001), held jointly with HCI International 2001. New Orleans, Loussiana USA. 5-10 August 2001.

[Heylighen, 96] Heylighen, F.; Bollen, J. "The World-Wide Web as a Super-Brain: from metaphor to model" R. Trappl (ed.). Cybernetics and Systems '96 (Austrian Society for Cybernetic Studies). Pp: 917. 1996.

[Heylighen, 99] Heylighen, F. "Collective Intelligence and its Implementation on the Web: algorithms to develop a collective mental map". Computational and Mathematical Organization Theory 5(3). Pp: 253-280.

[Hijikata, 01] Hijikata, Y.; Yoshida, T.; Nishida, S. "Adaptive Hypermedia System for Supporting Information Providers in Directing Users through Hyperspace". Proceedings of the 3rd Woorkshop on Adaptive Hypertext and Hypermedia. HYPERTEXT'01. Pp: 14-18. Arhus, Denmark. August, 2001.

[Höök, 98] Höök, H. “Evaluating the Utility and Usability of an Adaptive Hypermedia System”. Journal of Knowledge Based Systems. Volume 10, issue 5. 1998.

[Hübscher, 01] Hübscher, R. “What’s in a Prerequisite”. Int’l Conference on Advanced Learning Technology (ICALT 2001). Madison, WI, USA. 2001.

[IBM] http://www.ibm.com

[Iglesias, 03] Iglesias, A.; Martínez, P.; Fernández, F. “An Experience Applying Reinforcement Learning in a Web-Based Adaptive and Intelligent Educational System”. Informatics in Education. Volume 2. Pp: 1–18. Institute of Mathematics and Informatics, Vilnius. 2003.

[Java, 94] Java Development Kit. http://java.sun.com. Copyright 1994-2004 Sun Microsystems, Inc.

[Joslyn, 96] Joslyn, C. "Semantic Webs: A Cyberspatial Representational Form for Cybernetics". Proceedings of European Conference on Cybernetics and Systems Research. Ed. R. Trappl. Volume 2. Pp: 905-910. Vienna, 1996.

[Kozaczynsky, 92] Kozaczynsky, W.; Ning, J.; Engberts, A. “Program Concept Recognition and Transformation”. IEEE Trans.S.E. 18,12. Pp:1065-1075. 1992.

[Landow, 91] Landow, G.P.; Delany, P. “Hypertext, Hypermedia and Literary Studies: The State of the Art”. Hypermedia and Literary Studies, MIT Press. MA, Pp:3-50. Cambridge, 1991.

[Laszlo, 87] “Evolution: The Grand Synthesis”. New Science Library/Shambhala Publications, Inc. 1987. Trad. Es. “Evolution. La gran síntesis”. Madrid. Espasa-Calpe. 1998.

[Lehman, 00] Lehman, M.; Ramil, J. ”Towards a Theory of Software Evolution and its Practical Impact”. Principles of Software Evolution. IPSE 2000. IEEE Computer Society. Pp:2-13. 2000.

[LeMoigne, 77] Le Moigne, J. L. “La Théorie du Systtéme Général. Théorie de la Modélisation”. Presses Universitaries de France. París, 1977.

[LeMoigne, 90] Le Moigne, J. L. “La Théorie du Systtéme Général. Théorie de la Modélisation”. Presses Universitaries de France. Pp: 62. París, 1990.

[LeMoigne, 95] Le Moigne, J. L. “Les épistémologies constructivistes”. Paris. PUF. Que Sais-je ? N- 2969, 1995.

[Lindgreen, 90] Lindgreen, P. “A Framework of Information Systems Concepts”. Report IFIP WG8.1 TASK GROUP FRISCO. 1990.

[Lozano, 01] Lozano, A. “Ontologías en la Web Semántica”. I Jornadas de Ingeniería Web. Cáceres, España. 25-26, Junio, 2001.

[Maedche, 02] Maedche, A. “Ontology Learning for the Semantic Web”. Kluwer. 2002.

[Martín, 03] Martín Puente, E; Medina Medina, N; García Cabrera, L. "Educational Support in an Adaptive Hipermedia System: SEM-HP model". Proceedings of the 2nd international conference on multimedia ICT's in Education. M-ICTE'03. Volumen III. ISBN: 84-96212-12-2. Pp: 1642-1646. Badajoz, Spain. December 3-6, 2003.

[McDaid, 91] McDaid, J. "Breaking Frames: Hyper-Mass Media". Hypertext/hypermedia Handbook. Intertext Publications/MacGraw-Hill. Pp:445-458. New York, 1991.

[McGuiness, 04] McGuiness, D.; Van Harmelen, F. "OWL Web Ontology Language. Overview". http://www.w3.org/TR/owl-features. 2004.

[Medina, 02] Medina Medina, N.; García Cabrera, L.; Rodríguez Fórtiz, M.J.; Parets Llorca, J. "Adaptación al Usuario en Sistemas Hipermedia: El Modelo SEM-HP". II Jornadas de trabajo DOLMEN. Pp: 175-185. Valencia, 12 y 13 de Marzo de 2002.

[Medina, 02b] Medina Medina, N.; García Cabrera, L.; Torres Carbonell, J.J.; Parets Llorca, J. "Evolution in Adaptative Hypermedia Systems". Proceedings of the first international workshop on principles of software evolution. IWPSE 2002. Pp: 34-38. Orlando, Florida, USA. May 19-20, 2002.

[Medina, 02c] Medina Medina, N.; García Cabrera, L.; Rodríguez Fortiz, M.J.; Parets Llorca, J. "Adaptation in an Evolutionary Hyperpedia System: Using Semantic and Petri Nets". Adaptative Hypermedia and Adaptative Web-Based Systems (second international conference, AH 2002). Málaga, Spain. May29-31, 2002. Lectures Notes in Computer Science. Vol. 2347. ISSN: 0302-9743. Pp: 284-295. LNCS Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany.

[Medina, 02d] Medina Medina, N.; Molina Ortiz, F.; García Cabrera, L.; Rodríguez Fortiz, MJ. "Un Modelo para el Desarrollo de Sistemas Hipermedia Adaptativos". II Workshop de Investigación sobre nuevos paradigmas de interacción en entornos colaborativos. COLINE'02. Granada, 11 y 12 de Noviembre de 2002.

[Medina, 02e] Medina Medina, N.; García Cabrera, L.; Parets Llorca, J. "Taxonomía de Sistemas Hipermedia Adaptativos". Taller en sistemas hipermedia colaborativos y adaptativos. VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial, 18 de Noviembre de 2002.

[Medina, 02f] Medina Medina, N.; García Cabrera, L. "Modelos de Evolución en SEM-HP". III Jornadas de trabajo DOLMEN. Pp 71-77. VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial, 18 de Noviembre de 2002.

[Medina, 03] Medina Medina, N; Molina Ortiz, F.; García Cabrera, L; Parets Llorca, J. "Personalized Guided Routes in an Adaptive Evolutionary Hipermedia System". 9th International workshop on computer aided system theory". Eurocast'03. Selected Paper in Lectures Notes in Computer Science. Vol. 2809. ISSN: 0302-9743. ISBN: 3-540-20221-8. Pp:196-207. LNCS Editorial Springer-Verlag. Berlin Heidelberg, New York.

[Medina, 03b] Medina Medina, N.; Molina Ortiz, F.; Rodríguez Fortiz, MJ.; García Cabrera, L. "An Architecture for the Development, Evolution and Adaptation of Hipermedia Systems". Proceedings of the international conference on software

engineering research and practice. SERP'03. Volumen I. ISBN:1-932415-19-X. Pp:112-119. Las Vegas, Nevada, USA. June 23-26, 2003.

[Medina, 03c] Medina Medina, N.; García Cabrera, L.; Molina Ortiz, F. "Diversidad de Tipos de Navegación en un SHA". Taller en sistemas hipermedia colaborativos y adaptativos (2ª edición). Pp 1-9. VIII Jornadas de Ingeniería del Software y Bases de Datos. Alicante, del 12 al 14 de Noviembre de 2003.

[Medina, 03d] Medina Medina, N; Molina Ortiz, F.; García Cabrera, L; Rodríguez Fortiz, MJ. "Coevolution of Models of an Adaptive Hipermedia System". Proceedings of the 7th biennial world conference on integrated design and process technology. IDPT'03. ISSN: 1090-9389. Volumen II. Pp:18, 38-44. Austin, Texas, USA. December 3-6, 2003.

[Medina, 04] Medina Medina, N; Molina Ortiz, F.; García Cabrera, L. "A Hypermedia Model for an Adaptive Learning". *Proceedings of the IADIS international conference. e-Society 2004.* Volumen I. ISBN: 972-98947-5-2. Pp: 276-283. Ávila, Spain. July 16-19, 2004.

[Mens, 01] Mens, T. "Transformational Software Evolution by Assertions". Formal Foundations for the Evolution of Hypermedia Systems. 5th European Conference on Software Maintenance and Reengineering, Workshop on FFSE. IEEE Press. Pp: 67-74. Lisbon, Portugal. March 2001.

[Mens, 03] Mens, T.; Buckley, J.; Zenger, M.; Rashid, A. "Towards a Taxonomy of Software Evolution", USE 2003. Second International Workshop on Unanticipated Software Evolution. 18 pages. ETAPS'03. Warsaw, Poland. April 5 - 13, 2003. On.line http://joint.org/use2003/Papers/papers.html

[Molina, 02] Molina Ortiz, F.; García Cabrera, L.; Medina Medina, N.; Hurtado Torres, MV. "Editor de Estructuras Conceptuales Evolutivas: Consideraciones Prácticas". III Jornadas de trabajo DOLMEN. Pp 77-83. VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial, 18 de Noviembre de 2002.

[Molina, 03] Molina Ortiz, F.; Medina Medina, N.; García Cabrera, L. "JSEM-HP, una herramienta de autor para el desarrollo de sistemas hipermedia adaptativos evolutivos". IV Jornadas de trabajo DOLMEN. Pp 110-115. VIII Jornadas de Ingeniería del Software y Bases de Datos. Alicante, 11 de Noviembre de 2003.

[Morin, 77] Morin, E. "Le Methòde. Tome I: La Nature de la Nature". Ed. Du Suil. París, 1977. (Trad. Es. El Método I: La Naturaleza de la Naturaleza. Ed.Cátedra. Madrid, 1986).

[Murray, 99] Murray, T. "Authoring Intelligent Tutoring Systems: an analysis of the state of the art". International Journal on Artificial Intelligent in Education. Volume 10. Pp: 98-129. 1999.

[Nelson, 67] Nelson, T. H. "Getting in Out of Our System". In Schechter, G. Ed., Information Retrieval: A Critical Review, Thompson Books, Washington D.C., 1967.

[Nelson, 87] Nelson, T. H. "All for One and One for All". Hypertext'87. University of North Carolina, Chapel Hill, NC, v-vii. 1987.

[Netbeans, 00] Netbeans Integrated Development Environment. http://www.netbeans.com. Project Sponsored by Sun MicroSystems. Open source in June, 2000.

[Nielsen 90] Nielsen, J. "Hypertext and Hypermedia". Academic Press. On Navigation in Textual Virtual Environments and Hypertext". Pp: 155. 1990.

[Niertrasz, 00] Niertrasz, O.; Achermann, F. "Supporting Compositional Styles for Software Evolution. IPSE 2000. IEEE Computer Society. Pp: 14-22. 2000.

[Nilsson, 87] Nilsson, N.J. "Principios de Inteligencia Artificial". Ed. Diaz de Santos. 1987.

[Nilsson, 00] Nilsson, N.J. "Inteligencia Artificial: Una nueva síntesis". Ed. McGraw Hill. 2000.

[Not, 99] Not, E.; Zancanaro, M. "Reusing Information Repositories for Flexibly Generating Adaptive Presentations". IEEE International Conference on Information, Intelligence and Systems. Washington D.C. November, 1999.

[Not, 00] Not, E. Zancanaro, M. "The MacroNode Approach: Mediating Between Adaptive and Dynamic Hypermedia". International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH'2000). Trento. August, 2000.

[Parets, 93] Parets Llorca, J.; Anaya Morito, A.; Rodríguez Fórtiz, M.J.; Paderewsky Rodríguez, P. "A Representation of Software Systems Evolution Based on the Theory of the General System". Computer Aided Systems Theory, EUROCAST'93. Lecture Notes in Computer Science. Volumen 763. Pp: 96-109. Springer-Verlag, 1993.

[Parets, 95] Parets Llorca, J. "Reflexiones sobre el Proceso de Concepción de Sistemas Complejos. MEDES: un Método de Especificación, Desarrollo y Evolución de Sistemas Software". Tesis doctoral, 1995.

[Parets, 96] Parets Llorca, J.; Torres Carbonell, J. "Software Maintenance versus Software Evolution: An Approach to Software Systems Evolution". IEEE Conference and Workshop on Computer Based Systems. Pp. 134-141. Friedrichafen. March, 1996.

[PICS] http://www.w3.org/TR/rdf-pics

[Pressman, 87] Pressman, R. "Ingeniería del Software: un enfoque práctico". MacGraw-Hill, 1987.

[RDF] http://www.w3.org/RDF/

[Rodríguez, 01] Rodríguez Fortíz, MJ.; Paderewski Rodríguez, P.; García Cabrera, L.; Parets Llorca, J. "Evolutionary Modeling of Software System: its Application to Agent–Based and Hypermedia System". Internacional Workshop on Principles of Software Evolution. IWPSE'01. Pp. 54-61. Vienna, Austria. September 10-11, 2001.

[Rodríguez, 04] Rodríguez Fórtiz, M.J.; Paderewski Rodríguez, P.; Rodríguez Almendros, M.; Gea Mejías, M. "Unanticipated Runtime Adaptation of a Communicator for Autistic Children". Workshop FUSE 2004 (Foundations of

Unanticipated Software Evolution) in ETAPS'04 (European joint conferences on Theory And Practice of Software). Pp: 1- 15. Barcelona, 2004

[Russell, 96] Russell, S.; Norvig, P. "Inteligencia Artificial. Un enfoque moderno". Prentice Hall Hispanoamericana, S.A. 1996.

[Royce, 70] Royce, W.W. "Managing the development of large software systems". IEEE WESCON. Pp:1-9. August, 1970.

[Selker, 94] Selker, T. "COACH: A Teaching Agent That Learns," Communications of the ACM 37. No. 7, 9299. July, 1994.

[Shneiderman, 89] Shneiderman, B.; Kearsley, G. "Hypertext Hands-On!: An Introduction to a New Way of Organizing and Accessing Information". Addison Wesley, 1989.

[Smith, 04] Smith, M. "OWL Web Ontology Language. Guide". http://www.w3.org/TR/owl-guide. 2004.

[Sommerville, 88] Sommerville, I. "Ingeniería del software". Addison-Wesley Iberoamericana, 1988.

[Studer, 98] Studer, S.; Benjamins, R.; Fensel, D. "Knowledge Engineering: Principles and Methods". Data and Knowledge Engineering, 25, Pp: 161-197. 1998.

[Tirado, 01] Tirado Morueta, R.; Flores García, D. "Aspectos Críticos para el Diseño de Hipermedias en la Enseñanza". @gora digit@l. Revista científica electrónica. ISSN: 1577-9831. Nº 1. La Educación del Futuro, el Futuro de la Educación. 2001.

[Torres, 99] Torres Carbonell, J. Parets Llorca, J. "A Formalization of the Evolution of Software Systems". Computer Aided Systems Theory, EUROCAST'99. Pp: 269-272. Vienna. September, 1999.

[Torres, 02] Torres Carbonell, J. "Evolución de Sistemas Software. Aplicación de Modelos Biológicos a la Concepción Evolutiva de Sistemas Software". Tesis Doctoral. Granada. Mayo, 2002.

[Urretavizcaya, 01] Urretavizcaya, M. "Sistemas Inteligentes en el Ámbito de la Educación". Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.12. Pp: 5-12. 2001.

[Van Heijst, 97] Van Heijst, G.; Schrreiber, A. T.; Wielinga, B. J. "Using Explicit Ontologies in KBS Development". International Journal of Human and Computer Studies, 45, Pp:183-292. 1997.

[Vernet, 01] Vernet, D.; Salamó, M.; Vallespí, C.; Camps, J.; Golobardes, E.; Bacardit, J. "¿Cómo predecir la evolución del alumno?" Actas de las VII Jornadas de Enseñanza universitaria de la Informática. Jenui 2001. Palma de Mallorca, 16-18 Julio, 2001.

[Wadge, 01] Wadge, W.W.; Schraefel, M. C. "A Complementary Approach for Adaptive and Adaptable Hypermedia: Intensional Hypertext". Twelfth ACM

Conference on Hypertext and Hypermedia, Third Workshop on Adaptive Hypertext and Hypermedia. Aarhus, Denmark. August, 2001.

[Winston, 75] Winston, P. “Learning Structural Descriptions from Examples”. In P. Winston (Ed.), The Psychology of Computer Vision. Mcgraw-Hill.1975.

[Wenger, 87] Wenger, E. “Artificial Intelligence and Tutoring Systems”. Los Altos, CA, Morgan Kaufmann. 1987.

[Wermenlinger, 01] Wermenlinger, M.; Lopes, A.; Fiadeiro, J. L. ”A Graph Transformation Approach to Architectural Run-time Reconfiguration”. Formal Foundations for the Evolution of Hypermedia Systems. 5th European Conference on Software Maintenance and Reengineering, Workshop on FFSE. IEEE Press. Pp: 59-66. Lisbon, Portugal. March 2001.

[Unicode] Unicode HomePage. http://www.unicode.org/

[Wang, 98] Wang, W.; Rada, R. “Structured Hypertext with Domain Semantics”. ACM Trans. Information Systems. Volume 16(4). October, 1998.

[Woolf, 84] Woolf, B.; McDonald, D. “Building a Computer Tutor: Design Issues”. Computer IEEE. Pp: 61-73. September, 1984.

[W3C] World Wide Web Consortium. http://www.w3.org

[Wu, 00] Wu, H.; De Bra, P.; Aerts, A.; Houben, G.J. “Adaption Control in Adaptive Hypermedia Systems”. Proceedings of the AH2000 Conference, pp. 250-259, 2000. Lecture Notes in Computing Science. Volume 1892. Springer Verlag.

[Wu, 00b] Wu, H.; Houben, G.J.; De Bra, P. "Supporting User Adaptation in Adaptive Hypermedia Applications". Proceedings InfWet2000. Rotterdam, the Netherlands. 2000.

[XML] http://www.w3.org/XML