Universidad de Granada

JSEM-HP: Una Herramienta para el Desarrollo de Sistemas Hipermedia Adaptativos y Evolutivos

MEMORIA QUE PRESENTA

Fernando Molina Ortiz

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

DIRECTORAS

Lina García Cabrera y Nuria Medina Medina

Departamento de Lenguajes y Sistemas Informáticos

Editor: Editorial de la Universidad de Granada
Autor: Fernando Molina Ortiz
D.L.: GR 1887-2012
ISBN: 978-84-9028-186-4

Fernando Molina Ortiz
Tesis Doctoral
Universidad de Granada
2011

La memoria titulada “JSEM-HP: Una Herramienta para el Desarrollo de Sistemas Hipermedia Adaptativos y Evolutivos”, que presenta D. Fernando Molina Ortiz para optar al grado de doctor, ha sido realizada dentro del programa de doctorado “Métodos y Técnicas Avanzadas de Desarrollo de Software" del Departamento Lenguajes y Sistemas Informáticos de la Universidad de Granada bajo la dirección de las doctoras Lina García Cabrera y Nuria Medina Medina.

Granada, a de de 2011

El Doctorando Las Directoras

Fdo: Fernando Molina Ortiz Fdo: Lina García Cabrera Fdo: Nuria Medina Medina

Agradecimientos

A mis directoras: sin ellas esto habría sido imposible. Por su incansable apoyo. Por hacer que mantenga la ilusión en este trabajo cuando yo lo daba por perdido.

A Pepe, que será siempre mi mentor.

A Mª del Mar, por sus largas conversaciones y su ejemplo de cómo afrontar la vida. ¡Ánimo!

A mis compañeros Ana A., Mavi, Mª José, Patri, Francis, Miguel, Ana S., Perico, Juan Carlos, Rosana y otros que se me quedan en el tintero. Por apoyarme y hacerme sentir uno más cuando ni siquiera había terminado la carrera, y por seguir haciéndolo cuando ya no trabajo en la Universidad.

A mis amigos, por una amistad sin condiciones, y a mi familia, por hacer que mi infancia y sus buenos ratos sigan ahí como si fuera ayer. Así todo es más fácil.

A mis padres por estar siempre ahí, dándomelo todo pase lo que pase. A mi hermano, por estudiar en la butaca del hospital, y por hacer a veces de hermano mayor aunque sea el pequeño.

Gracias a Olga por ser como es y permitirme compartirlo. Por tejer tantas cosas juntos.

JSEM-HP: Una Herramienta para el Desarrollo de Sistemas Hipermedia Adaptativos y Evolutivos

MEMORIA

RESUMEN

Los sistemas hipermedia automatizan el principio de asociación de ideas para acceder a la información (Bush et al., 1945) y se han convertido en un poderoso mecanismo para organizar y navegar información, especialmente en la Web (Lowe y Hall, 1999). Sin embargo, su diseño y mantenimiento siguen siendo en la actualidad problemas abiertos ya que a menudo los hiperdocumentos presentan información desestructurada y desactualizada.

En consecuencia el modelo SEM-HP propone un proceso de diseño y una arquitectura que permite diseñar sistemas hipermedia semánticos (lo que repercute en una adecuada estructuración de los contenidos) y evolutivos (lo que permite que la información ofrecida por el sistema siempre se encuentre actualizada y que incluso se adapte a las necesidades concretas de cada usuario individual o grupo de usuarios). Las características del modelo, lo hacen especialmente interesante para el desarrollo de sistemas hipermedia educativos (Kibby et al., 1990).

La presente tesis ha permitido desarrollar una herramienta software capaz de afrontar el desarrollo (desde la perspectiva del autor) y la navegación (desde la perspectiva del usuario) de sistemas hipermedia adaptativos y evolutivos conforme al proceso de diseño y arquitectura especificados en el modelo SEM-HP.

La herramienta ha posibilitado la validación del modelo en campos de aplicación reales, fundamentalmente relacionados con la educación. Además ha permitido reflexionar sobre distintos aspectos conceptuales del modelo SEM-HP, como son: evolución del software, representación de la información mediante redes semánticas y modelado de usuario y adaptación; lo cuál ha sido fundamental para refinar y enriquecer el mencionado modelo.

1. INTRODUCCIÓN

En esta tesis se ha ampliado y validado el modelo SEM-HP (García, 2001) (Medina, 2004) para el desarrollo de sistemas hipermedia adaptativos y evolutivos. Para ello, se ha diseñado e implementado una herramienta de autor, denominada JSEM-HP (Molina et al., 2006), que permite trabajar directamente con el modelo y probarlo en situaciones reales, con el objetivo de comprobar la viabilidad de SEM-HP, tanto en la construcción evolutiva de sistemas hipermedia como en la adaptación al usuario (Brusilovsky, 1996).

En concreto, el desarrollo de JSEM-HP, con el que se ha realizado una implementación real de evolución del software, ha permitido hacer una reflexión

sobre los sistemas adaptativos enmarcándolos dentro de algo más ambicioso como es la evolución de un SISTEMA. También, ha proporcionado una visión más precisa y completa de los sistemas adaptativos y en particular de los modelos de usuario que deben gestionarse para lograr una adaptación efectiva. Esta visión analítica ha hecho posible la definición de una taxonomía específica para el modelado de usuario.

Se ha podido constatar la viabilidad de un proceso de diseño y construcción de un sistema hipermedia de forma incremental e iterativa y también de algunas de las técnicas de adaptación propuestas en SEM-HP.

De igual modo, ha posibilitado su aplicación a dominios concretos como son los sistemas basados en agentes, el desarrollo de mapas conceptuales, en la creación de ontologías para la integración de información en la empresa (Hurtado-Torres, 2002), o los entornos educativos.

Finalmente, la construcción de JSEM-HP ha extendido el modelo SEM-HP con nuevos conceptos que han permitido definir nuevas formas de adaptación y de navegación, tanto individuales como grupales.

Con objeto de poner de manifiesto estos resultados, la presente memoria se divide en varias secciones:

La primera de ellas (sección 2) se dedica a enmarcar este trabajo dentro de los sistemas software evolutivos y de los sistemas hipermedia adaptativos, poniendo de manifiesto los problemas abiertos que justifican el modelo SEM-HP. Asimismo, se describe brevemente el modelo teórico SEM-HP dado que el principal objetivo de esta tesis es validar y ampliar dicho modelo de forma práctica.

En la sección 3 se realiza una discusión de forma conjunta de los resultados obtenidos en los siete trabajos de investigación seleccionados. En concreto, se describe las aportaciones de cada trabajo en algunas de las siguientes temáticas: Evolución del software, modelado de usuario, proceso de diseño y construcción de sistemas hipermedia adaptativos, herramienta JSEM-HP y aplicación de JSEM-HP a diversos campos.

La sección 4 incluye otros resultados recogidos en otras publicaciones no seleccionadas o que están pendientes de publicación.

A continuación (la sección 5) se resumen los resultados y conclusiones que se han alcanzado en esta memoria y se enuncian los trabajos futuros que quedan abiertos.

Finalmente, la sección 6 incluye el listado completo de las contribuciones publicadas relacionadas con la herramienta de autor JSEM-HP.

2. PLANTEAMIENTO

La Web ha contribuido decisivamente a la popularización en todo el mundo de las técnicas hipermedia. De hecho, los sistemas web son un subconjunto concreto de los sistemas hipermedia (SHs) (Díaz et al., 2005). Sin embargo, la comunidad de hipermedia está continuamente manifestando las deficiencias e inconsistencias teóricas y prácticas de la Web. En este sentido, autores como (Bieber et al., 1997) afirman que los desarrolladores web no incluyen las características de alto nivel (nodos y enlaces tipificados, enlaces con atributos, consultas basadas en la estructura, inclusiones, etc.) que son parte de lo que ellos llaman la cuarta generación de hipermedia.

En (Berners-Lee et al., 2002) se plantea una extensión de la red actual, la Web semántica, donde "*computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning*". Aunque varios autores han propuesto modelos de referencia con el objetivo de garantizar el correcto diseño de los SHs (por ejemplo, el metamodelo Trellis de Stotts y Furuta (1989) o el modelo de Dexter de Halasz y Schwartz (1990)), la web, desde un punto de vista abstracto, se basa en un modelo básico (Rivlin et al., 1994) dividido en un *submodelo de datos* y un *submodelo de proceso*. Este modelo es general y flexible, pero incompleto. El *submodelo de datos* no define ninguna restricción para la determinación de la consistencia de la red de información y el *submodelo de proceso* es primitivo, ya que no detalla los mecanismos para definir la navegación.

Por tanto, se necesitan entornos que permitan el desarrollo de aplicaciones hipermedia basadas en un determinado modelo de hipermedia que caracterice la información y determine cómo se puede organizar, presentar, navegar y adaptar. El modelo debe tener mecanismos para caracterizar la información (estructuración semántica (Wang y Rada, 1998)), reestructurar la información y presentarla, navegarla y adaptarla de forma consistente (García et al., 2002).

Por último, la Web debe ser capaz adaptar la presentación de la hipermedia y la navegación a las características particulares de sus usuarios (De Bra et al., 2004). En este campo de investigación, diferentes modelos y arquitecturas de Sistemas Hipermedia Adaptativo (SHAs) han surgido. Como referencia, podemos citar el Adaptive Hypermedia Application Model (AHAM) (De Bra et al., 1998) y la Adaptive Hypermedia Architecture (AHA) basada en el mismo (De Bra et al., 2003) y el modelo Munich (Koch y Wirsing, 2002). Todas estas contribuciones tratan de proporcionar SHs con entornos más sofisticados, capaces de hacer frente a los cuatro grandes desafíos de la web: la búsqueda de información, el diseño de la navegación, la adaptación y el mantenimiento de la información de la red.

Con este mismo objetivo, se ideó el modelo SEM-HP en el que se basa la herramienta JSEM-HP. SEM-HP es un modelo SEMántico, Sistémico y Evolutivo

para Sistemas Hipermedia que en su última versión permite el desarrollo de sistemas hipermedia adaptativos.

2.1 Teoría de la evolución del software

En trabajos previos (García, 2001) (Medina, 2004) se ha determinado que se puede progresar en los desafíos web por medio de: la representación del conocimiento, una representación explícita de la semántica del dominio conceptual del SH y la separación de aspectos durante el proceso de desarrollo de un SH.

El proceso de desarrollo de un SH debe ser por naturaleza iterativo y evolutivo. Es necesario establecer mecanismos de cambio que permitan modificar y mejorar el SH. En la teoría de la evolución del software (Parets y Torres, 1996), un Sistema Software (SS) es un conjunto de sistemas que interactúan entre sí y con el entorno. Estos sistemas tienen una estructura que ofrece una determinada funcionalidad. El desarrollador (o autor) realiza cambios en el SS durante su construcción (modificando su estructura) pero también más tarde durante su vida funcional. Estos cambios que modifican la estructura y/o la funcionalidad del sistema pueden producir adaptaciones que garanticen la utilidad de la interacción del SS con el entorno cambiante.

En el proceso evolutivo de un SS, el desarrollador es un elemento muy importante, pues es el encargado de diseñar la capacidad de evolución del SS mediante las acciones evolutivas (ACE) que producen los cambios necesarios en el sistema. Otro elemento importante es el META-SISTEMA (Parets y Torres, 1996), que implementa dichas ACEs y permite la interacción entre el SS y el encargado de su mantenimiento. El SS sufre los cambios definidos en las ACEs cambiando su estructura y, por tanto su funcionalidad. Además, estos cambios pueden desencadenar nuevos cambios que deben ser propagados por las ACEs para garantizar la consistencia del SISTEMA.

En consecuencia, desde una perspectiva general, el SS puede evolucionar de dos maneras diferentes:

- Una evolución impulsada por el desarrollador.
- Una auto-evolución del SS derivada del uso del mismo.

El primer tipo de evolución (cambios estructurales) implica una intervención directa de los desarrolladores que impulsan los cambios en el sistema. El segundo tipo (cambios funcionales) se pueden realizar de manera automática en función de ciertos mecanismos definidos previamente por el desarrollador y el comportamiento del usuario actual, por lo tanto, la intervención de los desarrolladores en este segundo caso es indirecta. Podemos considerar la adaptación del usuario como un caso especial de este segundo tipo de evolución (Medina et al., 2002).

2.2 El Modelo SEM-HP

El trabajo propuesto en esta tesis tiene como antecedentes dos tesis doctorales (García, 2001) (Medina, 2004), en las cuales se define y especifica el modelo SEM-HP a distintos niveles (figura 1). La primera establece sus fundamentos como modelo de sistemas hipermedia semánticos y evolutivos, y la segunda lo completa incorporando aspectos de modelado y adaptación al usuario dentro del mismo marco semántico y evolutivo.

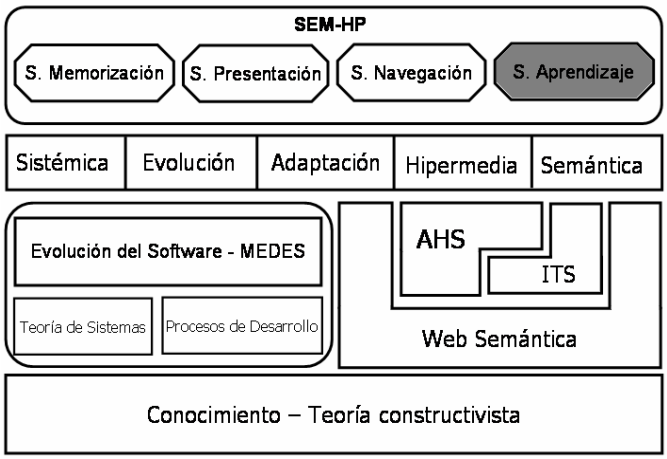


Figura 1: Marco de integración en SEM-HP.

Actualmente, el modelo SEM-HP (figura 2) proporciona al autor tres elementos para la creación de Sistemas Hipermedia Adaptativos (SHA): un proceso de desarrollo, una arquitectura y una herramienta de autor.

Figura 2: 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. Por su parte, la **herramienta de autor** facilita la creación del sistema de acuerdo a la arquitectura y el proceso de desarrollo propuestos en el modelo.

En concreto, la arquitectura del modelo SEM-HP establece una doble separación: a nivel vertical para distinguir cuatro aspectos funcionales y a nivel horizontal para distinguir dos niveles de abstracción.

La división vertical estructura un SHA en cuatro sistemas interrelacionados entre sí y en interacción: Memorización, Presentación, Navegación y Aprendizaje.

- El Sistema de Memorización (MS) define semánticamente, estructura y mantiene los conceptos y la información que ofrece el SHA.

- El Sistema de Presentación (PS) selecciona subconjuntos de los elementos ofrecidos por el MS, a los que se llama presentaciones.

- El Sistema de Navegación (NS) define las rutas o caminos que el usuario puede seguir cuando navega por cada presentación.

- El Sistema de Aprendizaje (LS) adapta la presentación y la navegación de los contenidos a las características específicas de cada usuario (adaptación al usuario). También analiza el comportamiento navegacional del grupo de usuarios (Medina et al., 2006) con objeto de sugerir ajustes evolutivos en los otros subsistemas (*feedback*).

Las características principales de estos cuatro subsistemas, implementados en la herramienta de autor JSEM-HP, pueden observarse en las tablas 1 a 4 que se muestran a continuación.

Tabla 1: Subsistema de Memorización de SEM-HP: Características principales.

Estructuración semántica del conocimiento	El Subsistema 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. 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.
Relaciones conceptuales	Las relaciones semánticas entre los conceptos se hacen explícitas en la estructura conceptual.
Relaciones funcionales	La función o rol que cada ítem de información juega respecto a cada concepto es descrito en la estructura conceptual etiquetando semánticamente la relación entre un ítem y cada concepto sobre el que trata su contenido.

Tabla 2: Subsistema de Presentación de SEM-HP: Características principales.

Reducción de complejidad	En la fase de presentación el autor selecciona distintos subconjuntos de la estructura conceptual de memorización, 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 más manejables. Así se consigue reducir los problemas de desorientación propios de la navegación sobre estructuras demasiado extensas.
Vistas parciales	El Subsistema de Presentación almacena varias presentaciones basadas en un mismo dominio de conocimiento. De alguna manera, establece los subdominios en que se divide el dominio de conocimiento capturado en el sistema. Se persigue con ello poder proporcionar al usuario una estructura de navegación centrada en la parcela de conocimiento que le interesa.

Tabla 3: Subsistema de Navegación de SEM-HP: Características principales.

Restricciones de navegabilidad	Las restricciones de navegabilidad determinan en qué sentido es navegable una relación conceptual. Por defecto, las relaciones conceptuales serán navegadas desde el concepto origen hasta el concepto destino. Sin embargo, se puede ampliar la navegabilidad de una relación conceptual en los dos sentidos siempre que sea conveniente hacerlo.

Ordenación parcial de la información	El Subsistema de Navegación establece un orden parcial a la hora de recorrer la información, basándose en las relaciones conceptuales existentes y en las restricciones de navegabilidad. Este orden permite contextos de lectura flexibles, pero se asegura que la información es accedida en un orden coherente con la semántica subyacente.

Tabla 4: Subsistema de Aprendizaje de SEM-HP: Características principales.

Aplicabilidad general	Permite desarrollar sistemas hipermedia adaptativos cualquiera que sea su dominio de conocimiento. De este modo, puede utilizarse para propósitos tan distintos como la creación de manuales educativos o la organización no jerárquica de documentos en un PC.
Adaptación integral	Proporciona numerosos métodos de adaptación para personalizar tanto el proceso de navegación como la presentación y el contenido de los documentos que se visitan.
Adaptación individual y colectiva	Los sistemas hipermedia desarrollados se adaptan a las características individuales de cada usuario, pero también recogen información sobre el proceso de navegación colectiva y, basándose en esta retroalimentación (análisis de matrices de transición), se identifican modificaciones estructurales que benefician al colectivo de usuarios (actual y futuro).
Adaptación al conocimiento	Proporciona herramientas para establecer 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.
Control sobre la adaptación	Otorga cierto grado de control al usuario sobre el tipo de adaptación del que desea disfrutar durante su proceso de navegación (admite cuatro modos de navegación con diferentes grados de adaptación) y sobre la actualización de su modelo de usuario.

Basándose en (Parets, 1995), la división horizontal de la arquitectura del modelo SEM-HP distingue dos niveles de abstracción: Sistema y Meta-sistema. El Meta-sistema proporciona al autor un conjunto de acciones evolutivas para que éste pueda realizar sobre cada sistema los cambios que considere oportunos, garantizándose que

tras éstos se mantiene la consistencia del sistema global. Para ello, establece un mecanismo basado en restricciones que verifica que los cambios introducidos son consistentes. Además, garantiza que el estado del sistema completo se mantiene consistente tras el cambio, mediante un proceso de evaluación y propagación tanto interna (dentro del subsistema modificado) como externa (dentro de los otros tres subsistemas).

Como se mencionó anteriormente, SEM-HP también propone un proceso de desarrollo con cuatro fases iterativas, cada una de las cuales se corresponde con uno de los cuatro sistemas mencionados. Excluyendo la fase de adaptación del usuario, las tres primeras fases son conceptualmente similares a los propuestos en otras metodologías, como son: Object Oriented Hypermedia Design Model (OOHDM) (Schwabe et al., 1996) o Web Modelling Language (WEBML) (Ceri et al., 2002).

2.3 Justificación del trabajo de tesis realizado

El modelo SEM-HP para el diseño de sistemas hipermedia evolutivos y adaptativos es una propuesta formal y sólida que especifica cómo construir herramientas hipermedia en el que el mantenimiento y la adaptación es posible gracias a la representación de la semántica y a la evolución. Como se ha visto antes, es fundamental la doble separación de aspectos, vertical, en cuatros sistemas (Memorización, Presentación, Navegación y Aprendizaje) y horizontal, en Sistema y Meta-sistema, para poder hacer evolucionar a los sistemas.

Quedaba por probar si realmente se podía llegar a desarrollar herramientas eficientes basadas en el modelo SEM-HP y si los SHA desarrollados con dichas herramientas eran usables y realizaban una adaptación efectiva. Por tanto, una vez completada la definición y especificación formal del modelo SEM-HP quedaban abiertos varios problemas importantes que justifican la presente tesis:

- De forma práctica, ¿se puede realizar un diseño y una implementación de la evolución en sistemas hipermedia?, en definitiva ¿podemos ofrecer una herramienta de autor basada en SEM-HP?

- ¿La evolución y la representación semántica/conceptual de un sistema hipermedia facilita el mantenimiento de un hiperdocumento?

- ¿La evolución en la que se basa el modelo SEM-HP posibilita la adaptación?

- ¿De qué forma podemos implementar el modelado de usuario para que un hiperdocumento pueda adaptarse de forma individual y grupal?, ¿qué características debe tener un modelo de usuario para que sea más o menos versátil en cuanto a la funciones de adaptación y posibilidades de evolución que ofrece?

- Finalmente, los sistemas hipermedia creados con una herramienta de autor basada en SEM-HP, ¿se pueden aplicar a dominios concretos como por ejemplo la docencia virtual?

Hasta ahora, no se ha podido constatar en la literatura especializada sistemas hipermedia adaptativos basados en modelos evolutivos. Sin embargo, el software evolutivo puede contribuir positivamente en el mantenimiento, la navegación y en la adaptación de los sistemas hipermedia. La construcción de la herramienta JSEM-HP es fundamental para reflexionar sobre el modelo SEM-HP y aplicarlo a campos concretos como el *e-learning*.

2.4 Objetivos de la tesis

El objetivo general de este trabajo es refinar y validar el modelo SEM-HP (García, 2001), (Medina, 2004) para el desarrollo de sistemas hipermedia adaptativos y evolutivos. Para ello, es del todo necesario el diseño e implementación de una herramienta de autor que permita trabajar directamente con el modelo y probarlo en situaciones reales, con la finalidad de comprobar las bondades de SEM-HP tanto en la construcción evolutiva de sistemas hipermedia como en la adaptación al usuario. Esto nos permitirá detectar problemas y/o identificar posibles mejoras o ampliaciones de SEM-HP. En concreto, los objetivos que pretende cubrir la presente tesis son los siguientes:

- Realizar un diseño abstracto de una arquitectura software que integre los principales elementos del modelo SEM-HP.

- Identificar y definir, si son necesarios, nuevos elementos en dicha arquitectura.

- Conforme al diseño realizado, desarrollar una herramienta software que implemente los principales elementos del modelo SEM-HP incluyendo dos partes interrelacionadas:

    - Una herramienta de autor que permita crear los sistemas hipermedia.

    - Un navegador que permita al usuario utilizar los sistemas creados.

- La herramienta desarrollada deberá ser fiel al carácter semántico, evolutivo y adaptativo del modelo SEM-HP.

- Utilizar la herramienta para validar el modelo, detectando posibles problemas e identificando y proponiendo mejoras o ampliaciones al mismo:

    - Mostrar de qué modo una herramienta basada en software evolutivo puede contribuir en los desafíos clave de los sistemas hipermedia: la búsqueda, la navegación, la adaptación y el mantenimiento.

- Se busca mostrar cómo los sistemas hipermedia basados en evolución amplían los modos de navegación y si efectivamente estas nuevas formas de navegación contribuyen a comprender mejor la información y a adquirir más fácilmente el conocimiento.

- Se pretende determinar si el diseño de la evolución facilita y ofrece nuevas formas de adaptación y qué características debe tener el modelado de usuario en estos casos.

- Se quiere constatar cómo un diseño evolutivo y una representación semántica de un sistema hipermedia hace posible el mantenimiento de los hiperdocumentos.

- Estudiar la aplicación del modelo y la herramienta a dominios concretos, como los entornos de docencia virtual o como la construcción de mapas conceptuales.

3. DISCUSIÓN CONJUNTA DE LOS TRABAJOS DE INVESTIGACIÓN PUBLICADOS

En esta sección se enumeran primero los trabajos seleccionados y, posteriormente se realiza una discusión conjunta de las aportaciones de estas publicaciones en las siguientes temáticas:

T1. Evolución del software.

T2. Modelado de usuario.

T3. Proceso de diseño y construcción de sistemas hipermedia adaptativos.

T4. Herramienta JSEM-HP.

T5. Aplicación de JSEM-HP a diversos campos.

3.1 Trabajos de investigación seleccionados

Trabajo 1: Applying Software Evolution Theory to Hypermedia Systems

Año: 2009

Molina Ortiz, F., García Cabrera, L., Medina Medina, N.

Int. J. Web Engineering and Technology (IJWET)

- La revista aparece indexada en: Scopus, Scirus, Google Scholar, ERGS, DBLP, COMPEN, COMDB, CIS, BUSASAP y ACM

- La revista tiene evaluación externa por pares y comité científico internacional.
- El artículo tiene 1 cita (no autocita) en Google Scholar
- El artículo tiene 19 páginas

Trabajo 2: Adaptation and User Modeling in Hypermedia Learning Environments using the SEM-HP Model and the JSEM-HP Tool

Año: 2010 (on-line), 2011 (impresa)

Medina Medina, N., **Molina Ortiz, F**., García Cabrera, L.

Knowledge and Information Systems

- El índice de impacto (Impact Factor) en JCR en el año 2010 es 2.000 La revista pertenece a la categoría: Computer Science, Information System, y ocupa la posición 24 de 116 en la categoría Computer Science, Information System. Por lo tanto, la revista se encuentra en el primer cuartil de Computer Science, Information System
- El artículo tiene 28 páginas

Trabajo 3: Diversity of Structures and Adaptive Methods on an Evolutionary Hypermedia System

Año: 2005

Medina Medina, N., **Molina Ortiz, F.**, García Cabrera, L.,

IEE Proceedings Software

- El índice de impacto (Impact Factor) en JCR en el año 2005 es 0.3
- La revista pertenece a la categoría: Computer Science, Software Engineering, y ocupa la posición 309 de 352 en la categoría Computer Science
- El artículo tiene 10 citas en Google Scholar, 7 citas en Scopus y 3 citas en ISI Web of Science (1 no autocita)

Trabajo 4: Applying a Semantic Hypermedia Model to Adaptive Concept Maps in Education

Año: 2007

Molina Ortiz, F., Medina Medina, N., García Cabrera, L.,

Lecture Notes in Computer Science

- La revista tiene evaluación por pares y comité científico internacional

Trabajo 5: JSEM-HP: A Tool for the Development of Adaptive Evolutionary Hypermedia Systems

Año: 2006

Molina Ortiz, F., Medina Medina, N., García Cabrera, L.,

Proceedings of the IADIS International Conference - Applied Computing 2006

- El artículo tiene 3 citas (2 no autocitas) en Google Scholar
- El artículo fue presentado en "IADIS International Conference - Applied Computing", clasificado con ranking C en CORE

Trabajo 6: An Author Tool Based on SEM-HP for the Creation and Evolution of Adaptive Hypermedia Systems

Molina Ortiz, F., Medina Medina, N., García Cabrera, L.,

Año: 2006

Workshop Proceedings of the Sixth International Conference on Web Engineering

- El artículo tiene 8 citas en Google Scholar (5 no autocitas)
- La revista aparece indexada en: Scopus, Google Scholar, DBLP y ACM

Trabajo 7: A Software System Evolutionary and Adaptive Framework: Application to Agent-Based Systems

Año: 2004

Paderewski Rodríguez, P., Torres Carbonell, J.J., Rodríguez Fórtiz, MªJ., Medina Medina, N., **Molina Ortiz, F.**

Journal of Systems Architecture

- El índice de impacto (Impact Factor) en JCR en el año 2004 es 0.242
- La revista pertenece a la categoría: Computer Science, Hardware & Architecture, y ocupa la posición 39 de 44 en la categoría Computer Science, Hardware & Architecture
- El artículo tiene 9 citas en Google Scholar (1 no autocita), 2 citas en ISI Web of Science y 3 citas en Scopus
- El artículo fue seleccionado después de ser presentado en la International Conference on Software Engineering Research and Practice, que ocupa la posición 108 de 735 en Computer Science Conference Ranking en el área Applications/Education/Software/Theory/Communications. Con índice de impacto 0,75 normalizado a 1

3.2 Discusión de los trabajos de investigación seleccionados

A continuación se reflejan de forma esquemática y colectiva las temáticas abordadas en cada trabajo de investigación seleccionado. Después, las subsecciones 3.2.1 a 3.2.5 describen y justifican más ampliamente la contribución de la tesis en cada temática, haciendo referencia a las aportaciones recogidas en los trabajos de investigación seleccionados.

	Trabajo						
Temática	**1**	**2**	**3**	**4**	**5**	**6**	**7**
T1: Evolución del software	√	√	√				
T2: Modelado de usuario	√	√	√			√	
T3: Proceso de diseño y construcción de sistemas hipermedia adaptativos	√	√		√	√	√	
T4: Herramienta JSEM-HP	√	√	√	√	√	√	
T5: Aplicación de JSEM-HP a diversos campos		√		√			√

3.2.1 Evolución del software

La implementación de la herramienta JSEM-HP nos ha permitido constatar los beneficios que conlleva un modelo que se basa en el software evolutivo y hasta qué punto es viable su construcción y qué mecanismos son clave en un enfoque evolutivo. Para ello, dos mecanismos fundamentales han sido: 1) una representación semántica explícita y 2) una implementación basada en la separación de aspectos.

Estos mecanismos nos permiten afrontar todos los desafíos asociados a los sistemas hipermedia: la búsqueda, la navegación, la adaptación y el mantenimiento. En

realidad el mantenimiento y la adaptación no son más que facetas distintas de la evolución. Considerar la adaptación como una forma de evolución ha hecho más uniforme y robusta su implementación.

Una semántica explícita permite búsquedas que son más precisas y automatizadas, ofrece navegadores conceptuales que facilitan la comprensión de los hiperdocumentos, mejora los modos de adaptación y de navegación y, por último, es fundamental para el mantenimiento.

Una representación e implementación de la semántica (basada en conceptos y relaciones entre conceptos y en conocimiento sobre esos conceptos) (figura 3) amplía las formas de navegación. Además de una navegación tradicional, se puede ofrecer una navegación basada en mapas o redes conceptuales, una navegación limitada por las restricciones conceptuales y una navegación basada en el conocimiento que los usuarios adquieren a medida que recorren la información del hipermedia.

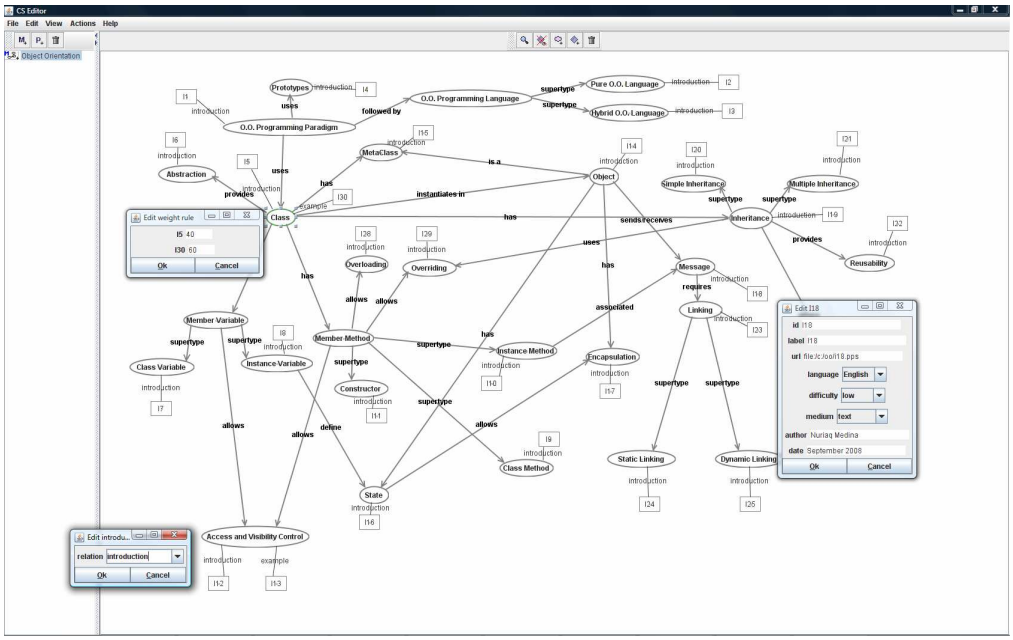


Figura 3: Desarrollo de una estructura conceptual de memorización en JSEM-HP.

La separación de aspectos facilita el desarrollo y el mantenimiento del sistema. El enfoque propuesto busca ser independiente de la plataforma tecnológica. La separación en sistemas y en niveles (sistema y meta-sistema) permite la clarificación del proceso de desarrollo y la verificación automática de los cambios y su propagación. Con JSEM-HP se ha podido validar cómo efectivamente ciertas acciones de evolución generadas en algunos de los sistemas son propagadas al resto manteniendo el sistema hipermedia en un estado consistente.

Publicaciones asociadas a esta parte:

Molina-Ortiz, F., García-Cabrera, L., Medina-Medina, N. Applying software evolution theory to hypermedia systems, Journal of Web Engineering and Technology, Vol. 5 (1), 69-87, 2009.

Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Adaptation and user modeling in hypermedia learning environments using the SEM-HP model and the JSEM-HP tool. Knowledge and Information Systems, 29 (3), 629–656, 2011 (DOI: 10.1007/s10115-010-0357-1).

Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Diversity of structures and adaptive methods on an evolutionary hypermedia system. Journal. IEE Proc.-Software. Vol.: 152 (3), 119-126, 2005.

3.2.2 Modelado de usuario

Durante el proceso de definición y diseño de JSEM-HP se adoptaron importantes decisiones sobre cómo modelar al usuario (figura 4). Para ello, se realizó un estudio y análisis de las diversas formas de modelado de usuario que son posibles en los sistemas adaptativos, en general, y en particular en los sistemas hipermedia adaptativos. El modelo de usuario es un elemento esencial ya que se puede registrar una gran variedad de información y dependiendo de esa información y el uso que se haga de ella se pueden llegar a ofrecer distintos modos de adaptación.

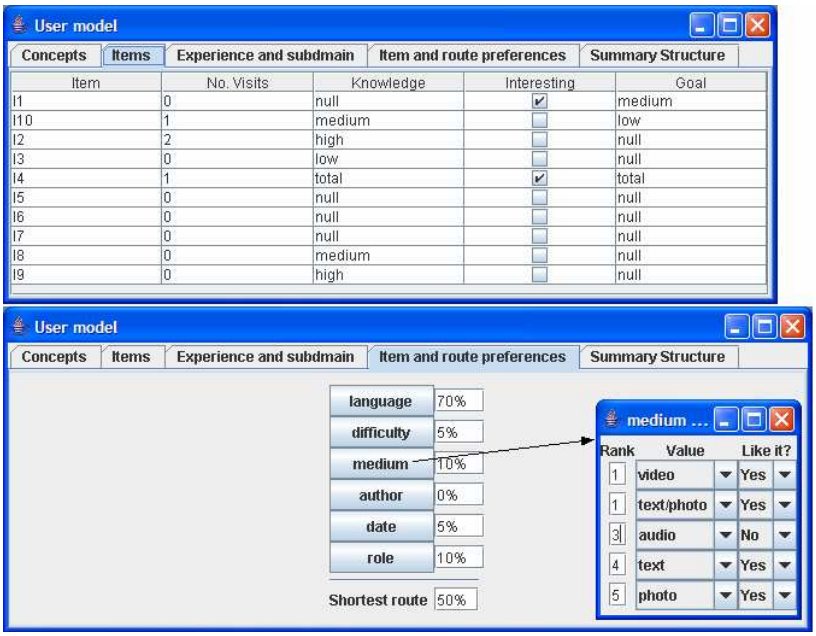


Figura 4: Modelo de usuario en JSEM-HP.

Se decidió aprovechar este estudio para realizar una taxonomía general para clasificar a los UMs desde múltiples perspectivas (granularidad, traza principal, la estructura, el contexto, inicialización, actualización, estabilidad, visibilidad, la dirección de la inferencia, etc.). Esta taxonomía ofrece una nomenclatura simple y un sistema de categorización que permite la comparación con otras propuestas afines de acuerdo con los criterios de clasificación establecidos en la taxonomía. Dicha taxonomía ha permitido caracterizar también el modelo de usuario de SEM-HP y valorar de forma más objetiva su potencial, detectando posibles mejoras que puede incorporar.

Además, con la herramienta JSEM-HP se ha podido constatar la viabilidad de algunas de las técnicas de adaptación al usuario previstas en el modelo SEM-HP como son: (1) la selección personalizada de la estructura de navegación (figura 5), (2) el poder ocultar y deshabilitar los enlaces a información inapropiada (figura 5) y (3) la anotación positiva de enlaces de interés (figura 5).

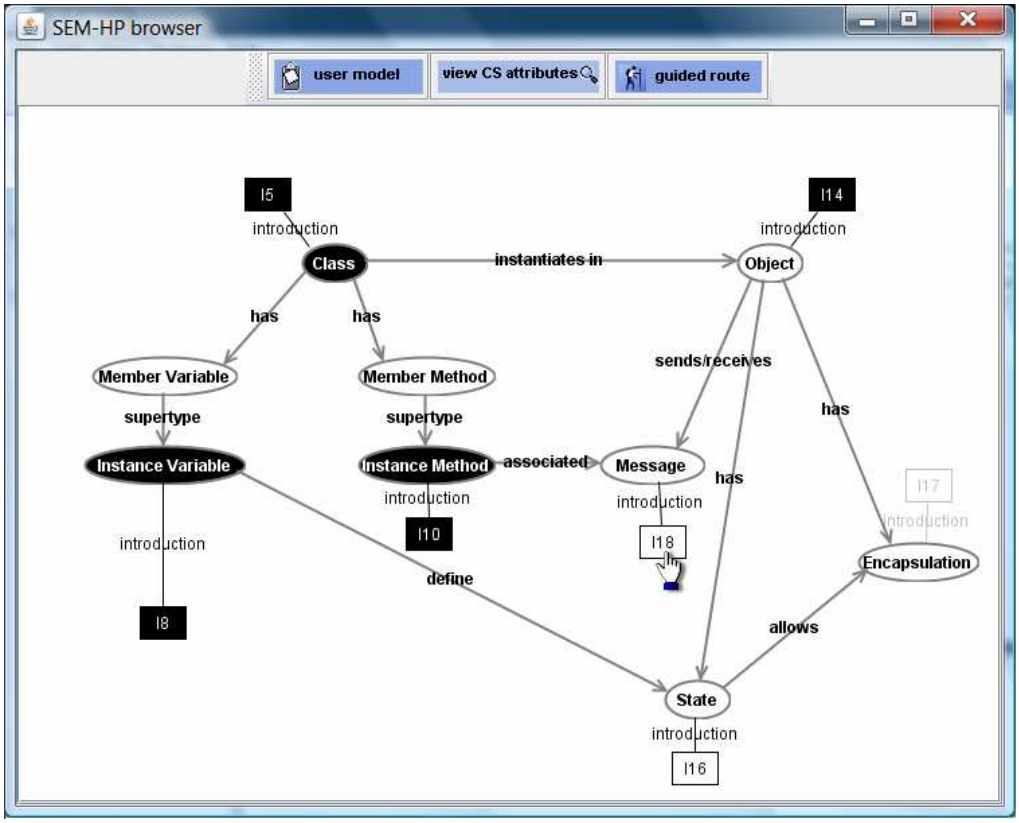


Figura 5: Adaptación en JSEM-HP.

Publicaciones asociadas a esta parte:

Molina-Ortiz, F., García-Cabrera, L., Medina-Medina, N. Applying software evolution theory to hypermedia systems, Journal of Web Engineering and Technology, Vol. 5 (1), 69-87, 2009.

Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Adaptation and user modeling in hypermedia learning environments using the SEM-HP model and the JSEM-HP tool. Knowledge and Information Systems, 29 (3), 629–656, 2011 (DOI: 10.1007/s10115-010-0357-1).

Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Diversity of structures and adaptive methods on an evolutionary hypermedia system. Journal. IEE Proc.-Software. Vol.: 152 (3),119-126, 2005.

Molina-Ortiz, F, Medina-Medina, N, García-Cabrera, L. An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems. First International Workshop on Adaptation and Evolution in Web Systems Engineering, ACM Press, 2006.

3.2.3 Proceso de diseño y construcción de sistemas hipermedia adaptativos

La implementación de la herramienta JSEM-HP ha respetado fielmente la arquitectura de doble dimensión (vertical y horizontal) propuesta en el modelo SEM-HP basada en la separación de aspectos. El uso de la herramienta ha demostrado que dicha arquitectura propicia un proceso de diseño incremental del sistema hipermedia y un control de los cambios en cualquiera de sus sistemas.

El autor del hiperdocumento puede añadir, modificar y eliminar conceptos y relaciones entre los mismos. Puede redefinir reglas de navegación y de adaptación. Todos estos cambios sólo se producen si se cumplen una serie de meta-restricciones. De igual modo, los cambios son propagados dentro y fuera de los sistemas, con el fin de mantener la consistencia global del sistema hipermedia.

Además, dicho proceso de diseño incremental permite diferentes estructuras de navegación y, por tanto, de modos de navegación. Finalmente, esta separación en subsistemas permite también un amplio abanico de formas de adaptación, tanto de forma previa a la navegación, posibilitado la selección de la presentación más cercana a las preferencias del usuario, como también durante la propia navegación, teniendo en cuenta el comportamiento que está teniendo el usuario en su interacción con el sistema.

Los numerosos ejemplos y ensayos parciales de uso de la herramienta han constatado la utilidad del modelo. Por otro lado, la aplicación de JSEM-HP a campos específicos, sobre todo los relacionados con la formación, han demostrado

que un diseño incremental e iterativo basado en la evolución facilita el mantenimiento, ofrece más posibilidades de navegación y de adaptación, y asiste en dicho proceso a autores y usuarios finales de los sistemas hipermedia.

Publicaciones asociadas a esta parte:

Molina-Ortiz, F., García-Cabrera, L., Medina-Medina, N. Applying software evolution theory to hypermedia systems, Journal of Web Engineering and Technology, Vol. 5 (1), 69-87, 2009.

Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Adaptation and user modeling in hypermedia learning environments using the SEM-HP model and the JSEM-HP tool. Knowledge and Information Systems, 29 (3), 629–656, 2011 (DOI: 10.1007/s10115-010-0357-1).

Molina-Ortiz, F., Medina-Medina, N., García-Cabrera, L. Aplying a semantic hypermedia model to adaptive concept maps in education. Computer Aided Systems Theory – EUROCAST 2007. Lecture Notes in Computer Science, 2007, Volume 4739/2007, 384-391, DOI: 10.1007/978-3-540-75867-9_49.

Molina-Ortiz, F, Medina-Medina, N, García-Cabrera, L. An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems. First International Workshop on Adaptation and Evolution in Web Systems Engineering, ACM Press, 2006.

Molina-Ortiz, F., Medina-Medina, N., García-Cabrera, L. JSEM-HP: A Tool for the development of adaptive evolutionary hypermedia systems. IADIS International Conference - Applied Computing, 595-599, 2006.

3.2.4 Herramienta JSEM-HP

JSEM-HP, es una herramienta de autor para el desarrollo de sistemas hipermedia. Ha sido implementada en Java por su portabilidad, la disponibilidad bibliotecas y por las facilidades para ser usado en Internet. Se basa en software libre, principalmente en Jgraph (Jgraph) para la manipulación de grafos.

JSEM-HP representa la semántica del dominio conceptual y de información, ofrece adaptación al usuario y permite un mantenimiento basado en la evolución. Los sistemas hipermedia creados con esta herramienta se basan en un tipo especial de red semántica, lo que da coherencia a la información ofrecida al usuario.

Durante la navegación (figura 6) la estructura conceptual que se ofrece al usuario se adapta de manera dinámica al conocimiento del usuario, por lo que no tendrá acceso a información que sea incapaz de comprender, y tendrá conciencia en todo momento de su proceso de aprendizaje (*modo de navegación por conocimiento*). Aunque si lo desea también puede elegir otros modos de navegación (Medina et al., 2005) con

dosis inferiores de adaptación (*modo de navegación por relación conceptual* y *modo de navegación por conceptos*), e incluso un modo libre de navegación (*modo de navegación tradicional*) que sería similar a navegar en la Web, pero usando una red semántica para seleccionar los ítems de información en lugar de únicamente enlaces embebidos en el texto.

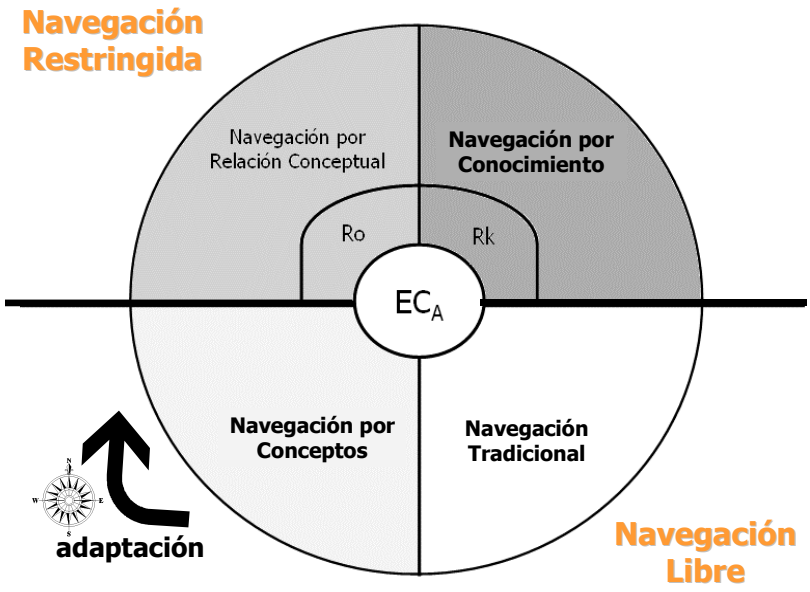


Figura 6: Modos de navegación permitidos en SEM-HP.

Por su parte, para poder implementar la evolución del sistema hipermedia, se ha definido de forma explícita la clase meta-sistema, que ejecuta las acciones de evolución que se pueden ejecutar desde cualquier sistema de SEM-HP. El metasistema se encarga de comprobar si la acción evolutiva dejaría al sistema en un estado consistente. Si es así, la lleva a cabo junto con otros cambios necesarios para conservar la consistencia (propagación del cambio), y en caso contrario la rechaza.

Cada tipo de acción de evolución se representa por una clase y los miembros son las acciones que se llevan a cabo. En el meta-sistema se ha preferido usar clases en vez de métodos por que facilita la notificación de cambio y la propagación de cambios. De igual modo, JSEM-HP permite implementar de forma sencilla la historia de los cambios estructurales del sistema, la lista de las acciones evolutivas. En esta herramienta, el meta-sistema emplea un mecanismo orientado a transacciones para modificar las estructuras conceptuales, de este modo una transacción se puede revertir si la post-condición no se cumple.

Dado que el meta-sistema se encarga de modificar el modelo, el controlador debe solicitar los cambios en el meta-sistema mediante el envío de las acciones de evolución. Después de evaluar y aceptar las acciones propuestas, el meta-sistema

llevará a cabo las modificaciones en el modelo, que, como es habitual en el patrón MVC, notificará a sus vistas para que se actualicen siempre que sea necesario.

Son pocas las herramientas conceptualmente parecidas a JSEM-HP. Por ejemplo, la arquitectura AHA! distingue entre modelo de dominio, de usuario y de adaptación. Sin embargo, su navegación es convencional, no conceptual, y aunque proporciona reglas para adaptar el modelo de usuario no tiene como SEM-HP una perspectiva evolutiva. En consecuencia, no comprueba si las reglas son inconsistentes.

También, se ha comparado JSEM-HP con herramientas más generales para la representación de conocimiento como Concilla (Nilsson y Palmr, 1999) y CmapTools (Cañas et al., 2004). Concilla no incluye técnicas de adaptación ni tampoco ofrece un mantenimiento evolutivo. Por su parte, CmapTools se basa en mapas conceptuales (que asocian conceptos formando proposiciones), y permite asociar recursos (piezas de información) a los conceptos. Estos mapas son similares a las estructuras conceptuales de SEM-HP. La diferencia es que mientras en SEM-HP se determina la función que desempeña el recurso (existe una interpretación semántica), en CmapTools se clasifica por su formato. CmapTools carece de un modelo formal y no incorpora mecanismos ni de evolución ni de adaptación.

Publicaciones asociadas a esta parte:

Todas las publicaciones seleccionadas (detalladas al inicio de la sección 3), a excepción de la última, difunden en mayor o menor grado aspectos de la herramienta de autor JSEM-HP. Aunque dos de estas publicaciones se centran de manera especial en describir con mayor detalle la citada herramienta:

Molina-Ortiz, F, Medina-Medina, N, García-Cabrera, L. An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems. First International Workshop on Adaptation and Evolution in Web Systems Engineering, ACM Press, 2006.

Molina-Ortiz, F., Medina-Medina, N., García-Cabrera, L. JSEM-HP: A Tool for the development of adaptive evolutionary hypermedia systems. IADIS International Conference - Applied Computing, 595-599, 2006.

3.2.5 Aplicación de JSEM-HP a diversos campos

La evaluación de la herramienta JSEM_HP ha sido otro de los objetivos perseguidos. Para ello, por ejemplo se han realizado hiperdocumentos usando la herramienta para describir el propio modelo SEM-HP, recoger información relativa a un tipo de yoga denominado *hatha-yoga* o explicar programación y diseño orientado a objetos a alumnos universitarios. La evaluación más formal y extensa ha sido precisamente esta última. Por un lado, se transcribió un mapa conceptual existente en la asignatura de programación orientada a objetos (una de las asignaturas de la titulación Ingeniería en Informática impartida en la Universidad de

Granada) para validar la riqueza semántica del modelo. Además, siendo éste el estudio más significativo, se ha utilizado JSEM-HP para crear un sistema hipermedia adaptativo en el dominio de la programación orientada a objetos (para la misma signatura mencionada anteriormente). En primer lugar se ha construido la estructura conceptual del dominio mediante mapas conceptuales para luego asociarles la información (páginas y recursos) (figuras 3 y 5).

La evaluación fue un éxito para los 36 aspectos de la herramienta JSEM-HP que se probaron (figura 7), se realizó mediante un cuestionario de opinión que los estudiantes cumplimentaron después de usar el sistema para aprender los conceptos generales de programación orientada a objetos.

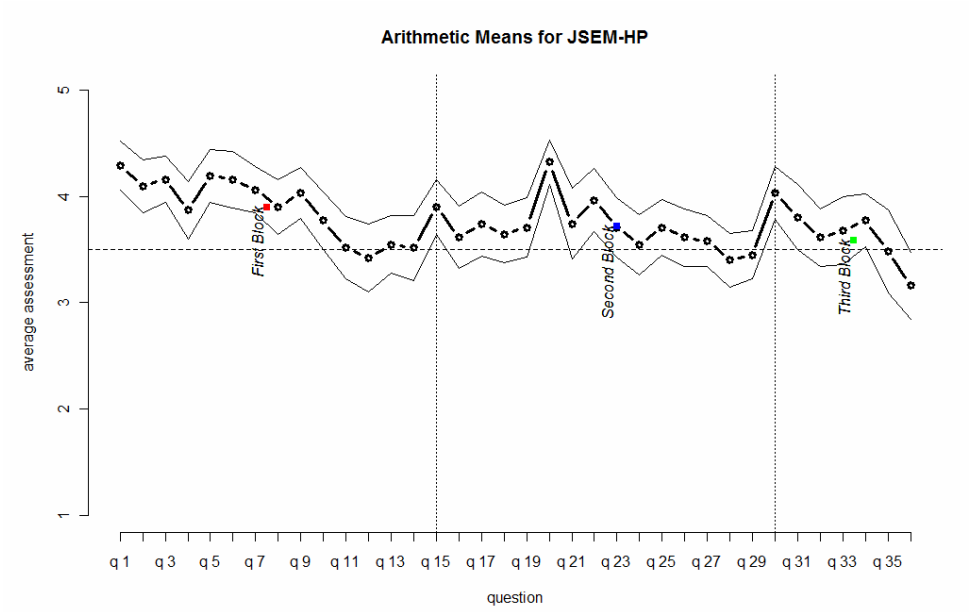


Figura 7: Evaluación de JSEM-HP.

Entre los principales resultado cabe destacar que, por ejemplo, los estudiantes dieron una puntuación media de 4,29 puntos (sobre 5) de la utilidad del navegador gráfico de JSEM-HP (en el modelo SEM-HP, la navegación se realiza a través de la red semántica en sí misma), 4,06 puntos a la utilidad de navegación en una visión parcial del grafo (el modelo SEM-HP proporciona al usuario una presentación diferente (CSP), según el subdominio del conocimiento en la que el usuario está más interesado), y 4,16 puntos a la adaptación y al modelado de usuario. También, se incluyeron en el cuestionario preguntas para comparar la navegación con JSEM-HP frente a la navegación tradicional de la Web. En este caso, los estudiantes dieron 3,8 puntos a la eficiencia de la búsqueda de información a través del navegador JSEM-HP en comparación con la búsqueda de información en la Web.

Publicaciones asociadas a esta parte:

Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Adaptation and user modeling in hypermedia learning environments using the SEM-HP model and the JSEM-HP tool. Knowledge and Information Systems, 29 (3), 629–656, 2011 (DOI: 10.1007/s10115-010-0357-1).

Molina-Ortiz, F., Medina-Medina, N., García-Cabrera, L. Aplying a Semantic Hypermedia Model to Adaptive Concept Maps in Education. Computer Aided Systems Theory – EUROCAST 2007 Lecture Notes in Computer Science, 2007, Volume 4739/2007, 384-391, DOI: 10.1007/978-3-540-75867-9_49

4. OTROS RESULTADOS

En esta sección describiremos otros resultados no recogidos en las publicaciones seleccionadas que, no obstante, también se han derivado de la tesis. Algunos de estas contribuciones han sido publicados en otros trabajos (ver sección 6) mientras que otros no se han difundido aún. Las aportaciones organizan en resultados conceptuales y resultados de implementación.

4.1 Resultados conceptuales

4.1.1 Dominio semántico

Durante la elaboración de la herramienta se ha definido el concepto de *dominio semántico*, publicado en (Molina et al., 2002). 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. Por ejemplo, una posible relación conceptual “hereda de”

tendría el mismo significado en todas las estructuras conceptuales asociadas a un dominio semántico de programación, mientras que una relación conceptual con el mismo nombre tiene un significado distinto si la asociamos a un dominio semántico de leyes o derecho, el cual sería común en todas las estructuras conceptuales asociadas a este dominio semántico.

4.1.2 Atributos genéricos

Para aumentar la versatilidad de la herramienta se han utilizado atributos genéricos (Molina et al., 2002). 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 obtiene 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*.

4.1.3 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 de añadir un concepto, el meta-sistema recibiría como parámetros la instancia correspondiente de *AddConcept* y 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 se definen con más facilidad según el estado del sistema después de efectuar la acción, y es muy difícil evaluarlas 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*.

4.1.4 Proceso de evolución

Tras la recepción de una acción evolutiva, el meta-sistema hace lo siguiente:

1. Comprueba las precondiciones de la acción. Si alguna no se cumple, la acción es rechazada.

2. Si es necesario, genera las acciones evolutivas necesarias para propagación interna del cambio que sea adecuado hacer antes de la acción evolutiva recibida, y las ejecuta.

3. Realiza la acción.

4. Comprueba las post-condiciones de la acción. Si alguna no se cumple, el sistema vuelve al estado anterior a la ejecución de la acción, y de la propagación previa del cambio, y la acción es rechazada.

5. Genera las acciones que implica la propagación interna (en el mismo sistema), en su caso, y las ejecuta.

6. Genera las acciones que implica la propagación externa (en los otros sistemas). Informa al autor sobre los sistemas afectados y ejecuta las acciones de propagación externa.

En este proceso se ha introducido un cambio con respecto al modelo SEM-HP, en cuya definición teórica la propagación del cambio es siempre llevada a cabo después de la ejecución de la acción evolutiva.

En la práctica hay veces que es mejor hacer la propagación del cambio antes de ejecución de la acción evolutiva, preparando al sistema para que quede consistente al ejecutar la acción evolutiva, y, en su caso, el resto de la propagación del cambio correspondiente. Por ejemplo, una vez eliminado un concepto se pierde el acceso a sus reglas de peso, por lo que es mejor borrar las reglas de peso antes de eliminar el concepto. En cambio, otras veces es mejor llevar a cabo la propagación del cambio después de ejecutar acción evolutiva, por ejemplo es más fácil buscar conceptos desconectados del concepto raíz después de eliminar un concepto o una asociación conceptual, ya que se puede utilizar sin cambios un algoritmo estándar. Debido a esto, la implementación distingue entre **propagación del cambio antes de ejecutar al acción evolutiva y después**, con las modificaciones pertinentes en las acciones evolutivas afectadas.

4.1.5 Uso de la herramienta de autor

En algunos casos se ha observado que la necesidad de mantener la consistencia en todo momento puede resultar demasiado rígida para el autor. Por ejemplo, la eliminación de un concepto o asociación conceptual puede acarrear la eliminación de muchos otros conceptos si éstos quedan desconectados del concepto raíz, y estamos hablando sólo de la propagación interna del cambio, que al menos es visible directamente: también se llevaría a cabo una propagación externa del cambio que puede no ser obvia para un autor que no esté muy familiarizado con SEM-HP. Esto puede resultar confuso para el autor, por lo que se ha considerado necesario buscar una solución que permita usar la herramienta a todo tipo de autores.

Por un lado, se debería informar al autor de la propagación de cada cambio (la implementación será sencilla, ya que se conocen las acciones de propagación del cambio e incluso éstas cuentan con un método para mostrarlas de forma legible). Esta información debería variar según el conocimiento que el autor tenga del modelo SEM-HP, para no aburrir a usuarios experimentados ni dejar de dar información útil a usuarios noveles. Se abre aquí otra posibilidad de adaptación al usuario, destinada en este caso al autor.

También es necesario un mecanismo eficiente para deshacer cambios. Por ahora se puede usar el mecanismo orientado a transacciones, pero la implementación actual puede no ser suficiente debido a los recursos que podría llegar a consumir si se guarda una historia muy larga. En el futuro se trabajará en un sistema orientado transacciones que requiera menos recursos o en un mecanismo que permita deshacer individualmente cada acción evolutiva.

Además, se puede optar por relajar en algunos casos la necesidad de que el modelo sea consistente en todo momento, lo que permitiría, por ejemplo, eliminar un concepto de forma que parte de la estructura conceptual correspondiente permanezca desconectada del concepto raíz, siempre y cuando la restricción se cumpla en un momento posterior, por ejemplo al cambiar a otra estructura conceptual o al grabar el modelo. Esto requeriría hacer un estudio de SEM-HP y de JSEM-HP para identificar en qué momentos se puede permitir que el sistema esté en un estado inconsistente, y cuándo y de qué manera se comprobará de nuevo la consistencia y se forzará que el sistema vuelva a ser consistente.

4.2 Resultados de implementación

4.2.1 Estructuras conceptuales

El diagrama de la figura 8 muestra las interfaces y clases principales de la herramienta 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.

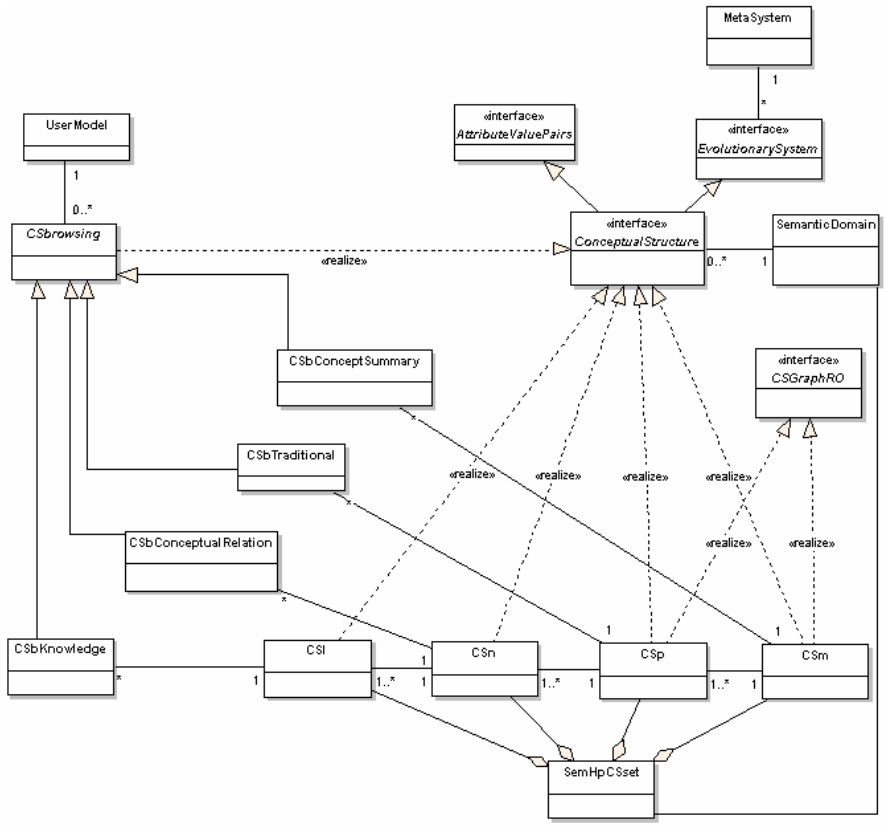


Figura 8: Estructuras conceptuales.

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 interfaz *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.

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.

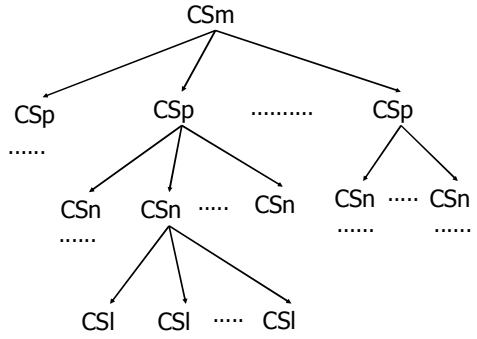


Figura 9: SemHpCSset

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 9), 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.

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 Expresiones lógicas

En el siguiente diagrama (figura 10) se muestra la estructura utilizada en la herramienta JSEM-HP para construir y modificar expresiones lógicas complejas necesarias para definir las reglas de navegación y adaptación.

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.

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 $K(i_j)$.

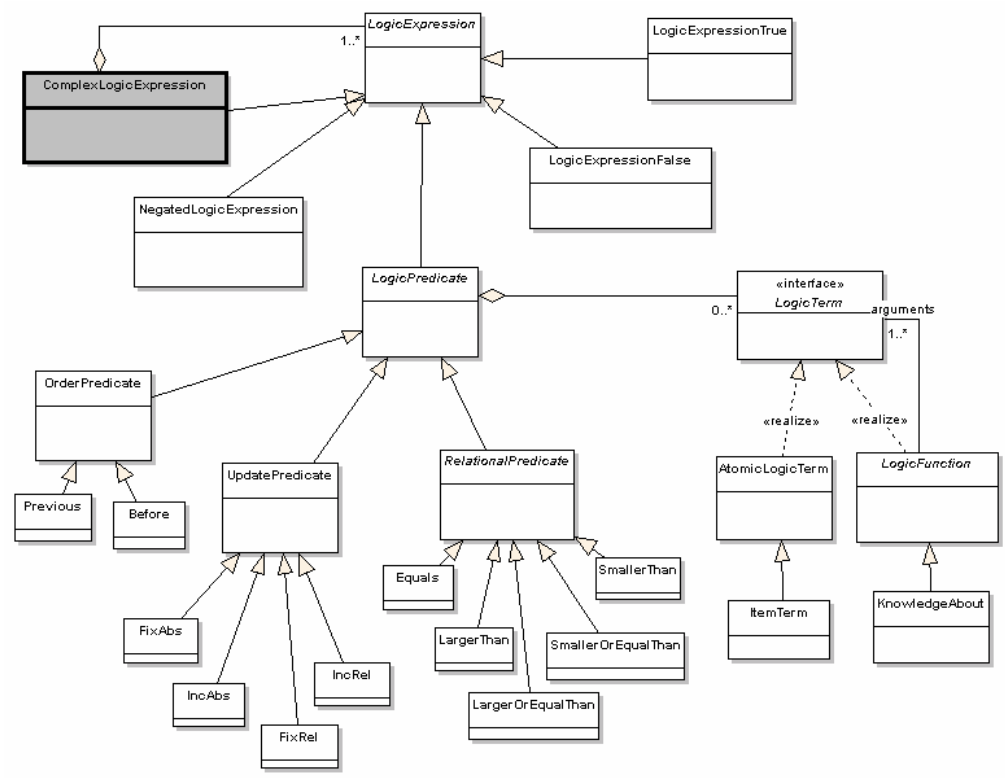


Figura 10: Expresiones lógicas.

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.

4.2.3 CSmDefault, CSpDefault, CSnDefault y CSlDefault

Aunque anteriormente 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 modo de ejemplo, la figura 11 muestra los métodos definidos en la interfaz que representa la estructura conceptual del sistema de memorización (CSm) y los métodos implementados en la clase que realiza dicha interfaz (CSmDefault) que realiza ésta y otras interfaces.

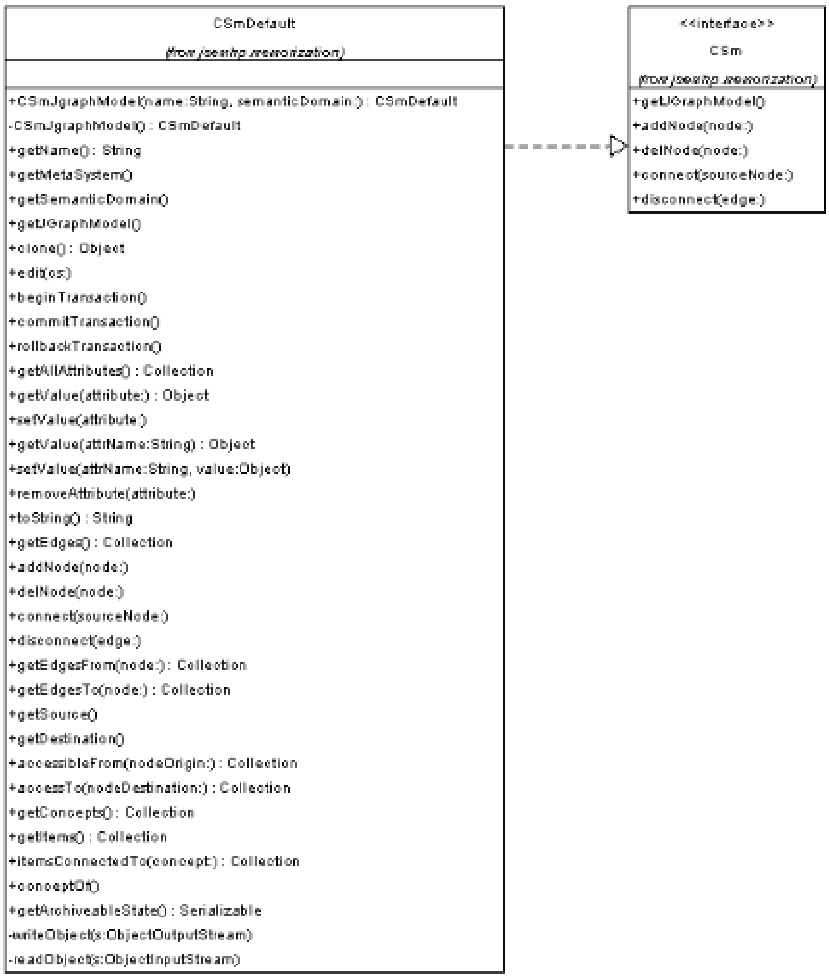


Figura 11: Métodos de *CSm*.

4.2.4 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 12).

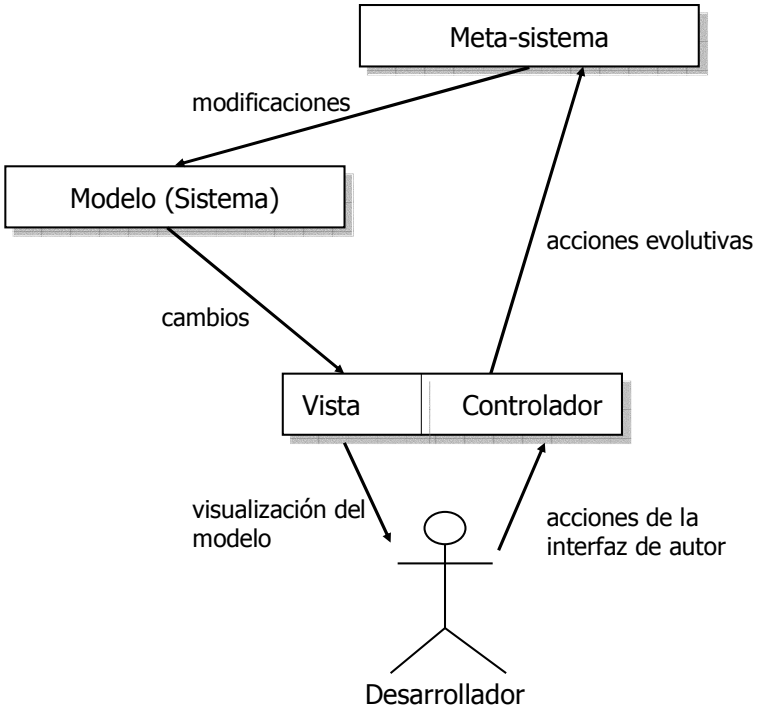


Figura 12: Modelo-vista-controlador adaptado al meta-sistema.

En la figura 13 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.

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 14.

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.

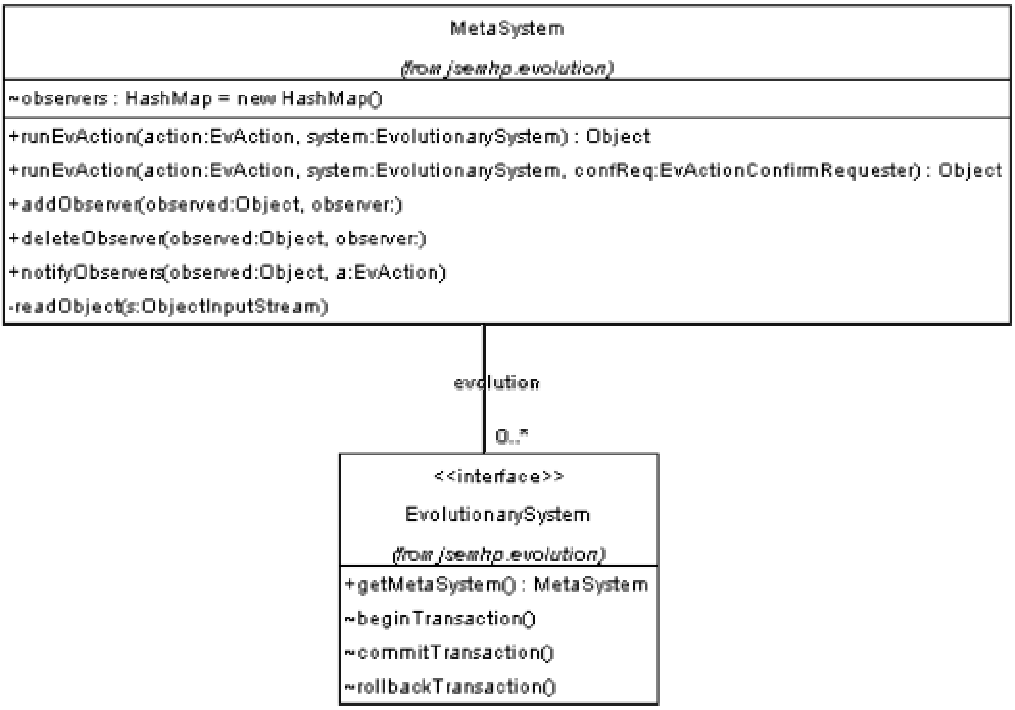


Figura 13: Sistema y Meta-Sistema.

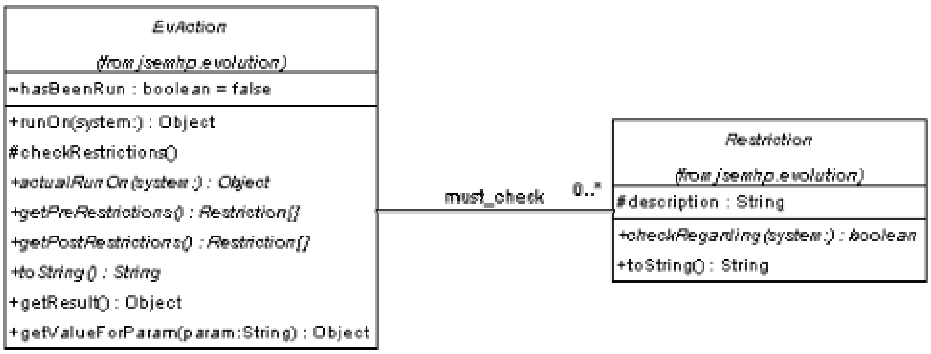


Figura 14: Acciones evolutivas y restricciones.

5. CONCLUSIONES Y PERSPECTIVAS

5.1 Conclusiones

Con el diseño e implementación de la herramienta de autor JSEM-HP, el modelo SEM-HP ha dejado de ser una propuesta teórica y formal para ser una especificación práctica de cómo pueden construirse herramientas hipermedia evolutivas y adaptativas. Adicionalmente, el desarrollo de JSEM-HP ha permitido: 1) disponer de una aplicación software para la creación, mantenimiento y uso de sistemas hipermedia adaptativos de propósito general y 2) reflexionar sobre distintos aspectos conceptuales del modelo SEM-HP, así como refinarlo y enriquecerlo y aplicarlo a campos concretos.

De forma más detallada, se resumen en esta sección las principales aportaciones que se pueden colegir de este trabajo:

- Se ha desarrollado una herramienta software que incluye tanto una herramienta de autor para crear sistemas hipermedia como un navegador con el que el usuario final puede recorrer de forma personalizada los hipermedia creados.

    - La implementación ha sido realizada siguiendo un enfoque de orientación a objetos puro, lo que significa que todo ha sido tratado como un objeto, incluido: el meta-sistema, las estructuras conceptuales, las expresiones lógicas, las acciones evolutivas, las restricciones de las acciones evolutivas, etc.

    - La implementación ha sido realizada persiguiendo una reutilización máxima, para ello se han incluido importantes conceptos como el de *atributo genérico* y *dominio semántico*. Se han definido interfaces que luego son implementadas de forma concreta en las clases y se ha seguido el patrón modelo-vista-controlador para la evolución.

- Durante el diseño de JSEM-HP se ha podido constatar que en un enfoque evolutivo hay dos mecanismos que son primordiales: la representación semántica/conceptual explícita y la implementación basada en la separación de aspectos.

    - Una semántica explícita permite búsquedas más precisas y automatizadas, amplia los modos navegación y las formas de adaptación y, por último, automatiza parcialmente el mantenimiento.

    - La separación de aspectos hace incremental y controlable el desarrollo y el mantenimiento. La separación en sistemas y en niveles clarifica el proceso de desarrollo y permite verificar los cambios llevando a cabo sólo los que dejan al sistema en un estado consistente. En JSEM-HP el autor puede de

forma dinámica cambiar la estructura conceptual y las reglas de navegación y de adaptación siendo asistido y controlado por la herramienta (impidiendo que pueda hacer cambios que no garanticen la integridad de sistema hipermedia).

- El uso de la herramienta ha permitido analizar de forma práctica las diversas formas de modelado de usuario en sistemas adaptativos; posibilitando la especificación de una taxonomía de modelado de usuario. Esta taxonomía (Medina et al., 2011) permite clasificar los modelos de usuario desde la perspectiva de su estructura y gestión; y ha sido usada para caracterizar el modelado de usuario en JSEM-HP (y para compararlo con otras propuestas afines).

  - Dicha caracterización a nivel estructural establece que el modelo de usuario en SEM-HP tiene granularidad individual y de grupo, trabaja con modelos "*overlay*" y estereotipos para representar internamente a los usuarios, su traza principal es el conocimiento adquirido por el usuario pero también registra sus preferencias e intereses, usa como modelo conceptual una red semántica, es un modelo episódico basado en parejas (atributo=valor) pero también usa matrices de transición para llevar a cabo las adaptaciones de granularidad gruesa, etc.

  - Dicha caracterización a nivel de gestión (inicialización, actualización y uso) establece que el modelo de usuario en SEM-HP se actualiza de forma dinámica pero también puede hacerse de forma explícita por parte del usuario, sigue un modelo predictivo para adaptaciones estructurales basadas en análisis estadísticos de las matrices de transición, es abierto para que los usuarios puedan ver/modificar su contenido, realiza inferencias *push* para producir la adaptación, etc.

- El desarrollo y uso de la herramienta ha permitido analizar y refinar distintos aspectos del modelo SEM-HP relacionados con la evolución y la adaptación al usuario.

  - En JSEM-HP se ha incorporado la filosofía de navegación multimodal definida en el modelo SEM-HP (navegación por conceptos, por relación conceptual, por conocimiento y libre). Principalmente, se ha trabajado con la navegación tradicional o libre y la navegación por conocimiento, permitiendo constatar su utilidad para los diferentes propósitos que cada modo tiene como objetivo cubrir.

  - Con JSEM-HP se han podido probar algunas de las técnicas de adaptación al usuario: la selección personalizada de una presentación, mostrar/ocultar enlaces teniendo en cuenta el comportamiento del usuario (principalmente su conocimiento) y la anotación de enlaces de interés.

- En JSEM-HP se han implementado los mecanismos evolutivos necesarios para desarrollar y mantener el sistema hipermedia adaptativo de una forma fácil, íntegra y consistente. Para poder implementar la evolución en JSEM-HP se ha definido de forma explícita la clase meta-sistema para que ejecute la acciones de evolución (comprueba precondiciones para rechazar el cambio y poscondiciones para poder volver al estado anterior). Si lleva a cabo la acción de evolución genera las acciones que implican tanto la propagación interna como externa.

- El diseño e implementación de JSEM-HP ha permitido ampliar y modificar el modelo SEM-HP.

  - Se ha añadido el concepto de *dominio semántico* que permite definir áreas de conocimiento dónde el significado de una etiqueta está bien determinado, posibilitando la reutilización de estas etiquetas y su significado en distintas redes semánticas.

  - Se ha llegado a la conclusión de que la propagación del cambio en algunos casos es mejor hacerla antes de la acción evolutiva, en lugar de ejecutarla después como inicialmente se dispuso en el modelo.

- La evaluación de la herramienta de autor ha permitido identificar algunas dificultades para autores que no estén familiarizados con el modelo SEM-HP (derivadas de la necesidad del modelo de mantener el sistema consistente en todo momento), proponiéndose mecanismos para solucionarlas o reducirlas.

- Finalmente, se ha evaluado la herramienta JSEM-HP desarrollando con ella diversos hiperdocumentos. Se ha aplicado con éxito en la enseñanza y en la construcción de mapas conceptuales.

5.2 Trabajos futuros

Se enumeran en este apartado aspectos que han quedado sin implementar del modelo SEM-HP, mejoras y problemas abiertos que pueden dar lugar a nuevas líneas de investigación:

1. Conseguir la interoperabilidad de JSEM-HP, para lo cual estamos considerando utilizar grafos estándar para los mapas conceptuales, como GXL (Graph eXchange Language) (Holt et al., 2000) y XTM (XML Topic Maps) (Colmenero, 2005).

2. Utilizar estándares para implementar el modelo de usuario garantizando así su portabilidad.

3. Incluir en el modelo de usuario aspectos contextuales y psicológicos. La inclusión de contexto del usuario (incluyendo el contexto psicológico) tanto en la información y la información de meta-contextos hace que sea posible proporcionar la información más útil para el usuario en ambos contextos. Con este propósito en mente, los autores están estudiando diversas técnicas de recuperación de información contextual y las metodologías de evaluación (Tamine-Lechani et al., 2009).

4. Completar la herramienta para que incorpore partes adicionales del modelo SEM-HP: completar el mantenimiento de la consistencia en todas las situaciones, terminar la implementación de los modos de navegación por conceptos y por relación conceptual e implementar de forma completa las rutas guiadas y la adaptación a grupos de usuarios.

5. Hacer que la herramienta sea accesible para usuarios con menos conocimientos de SEM-HP:

    - Mejorar la interfaz gráfica de JSEM-HP para que se más usable, más intuitiva y más dinámica, tanto desde la perspectiva de autoría como de navegación.

    - En el caso específico de los autores, hacer que la propagación del cambio confunda menos al autor en algunos casos, por un lado informando de sus efectos si es necesario (lo cual puede abrir un campo de adaptación al autor), y por otro implementando un mecanismo más eficiente para deshacer acciones. Se debe sopesar la posibilidad de permitir que el modelo esté inconsistente durante períodos cortos.

6. Fomentar el uso de la herramienta primero en la docencia impartida por el departamento de Lenguajes y Sistemas Informáticos y después extendiendo su uso en otros ámbitos afines.

7. Aplicar las estructuras conceptuales de JSEM-HP para mejorar la comprensión semántica de personas con problemas cognitivos, como puede ser el caso de personas con pérdidas de memoria temporales provocados por traumatismos craneales leves o personas con pérdidas progresivas de memoria por enfermedades como el *Alzheimer*.

6. LISTADO COMPLETO DE CONTRIBUCIONES

6.1 Capítulos de libro

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L., Parets Llorca, J.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 4 (2)

TÍTULO DEL LIBRO: Cast and Tools for Complexity in Biological, Physical and Engineering Systems - Extended Abstract

TÍTULO DEL CAPÍTULO: Personalized Guided Routes in an Adaptive Evolutionary Hypermedia System

CLAVE: Capítulo de Libro

COLECCIÓN: Eurocast

PÁGINAS: Desde: 135 Hasta: 138

EDITORIAL: Roberto Moreno-Díaz jr. et al. (Eds.)

CIUDAD: Las Palmas de Gran Canaria

PAÍS DE PUBLICACIÓN: España

AÑO DE PUBLICACIÓN: 2003

ISBN: 84-688-0820-2

INDICIOS DE CALIDAD:

- El artículo tiene 3 citas en Google Scholar (1 no autocita)

AUTORES: Paderewski Rodríguez, P., Torres Carbonell, J. J., Rodríguez Fórtiz, Mª J., Medina Medina, N., Molina Ortiz, F.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 5 (5)

TÍTULO DEL LIBRO: Proceedings of the International Conference on Software Engineering Research and Practice

TÍTULO DEL CAPÍTULO: A Software System Evolutionary and Adaptive Framework: Application to Agent-Based Systems

CLAVE: Capítulo de Libro

VOLUMEN: 1

COLECCIÓN: SERP

PÁGINAS: Desde: 142 Hasta: 148

EDITORIAL: CSREA Press. Ban Al-Ani, H. R. et al.(Eds.)

PAÍS DE PUBLICACIÓN: Estados Unidos de América (los)

AÑO DE PUBLICACIÓN: 2003

ISBN: 1-932415-19-X

INDICIOS DE CALIDAD:

- El artículo fue presentado en International Conference on Software Engineering Research and Practice, que ocupa la posición 108 de 735 en Computer Science Conference Ranking en el área Applications/Education/Software/Theory/Communications. Con índice de impacto 0,75 normalizado a 1

- El artículo tiene 9 citas en Google Scholar (1 no autocita)

AUTORES: Medina Medina, N., Molina Ortiz, F., Rodríguez Fórtiz, Mª J., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 4 (2)

TÍTULO DEL LIBRO: Proceedings of the International Conference on Software Engineering Research and Practice

TÍTULO DEL CAPÍTULO: An Architecture for the Development, Evolution and Adaptation of Hypermedia Systems

CLAVE: Capítulo de Libro

VOLUMEN: 1

COLECCIÓN: SERP

PÁGINAS: Desde: 112 Hasta: 119

EDITORIAL: CSREA Press. Ban Al-Ani, H. R. et al.(Eds.)

PAÍS DE PUBLICACIÓN: Estados Unidos de América (los)

AÑO DE PUBLICACIÓN: 2003

ISBN: 1-932415-19-X

INDICIOS DE CALIDAD:

- El artículo fue presentado en International Conference on Software Engineering Research and Practice, que ocupa la posición 108 de 735 en Computer Science Conference Ranking en el área Applications/Education/Software/Theory/Communications. Con índice de impacto 0,75 normalizado a 1.

- El artículo tiene 2 citas en Google Scholar y 2 citas (1 no autocita) en Scopus

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (2)

TÍTULO DEL LIBRO: Proceedings of the IADIS Internacional Conference e-Society

TÍTULO DEL CAPÍTULO: A Hypermedia Model for an Adaptive Learning

CLAVE: Capítulo de Libro

VOLUMEN: 1

COLECCIÓN: IADIS

PÁGINAS: Desde: 276 Hasta: 283

EDITORIAL: IADIS Press. Pedro Isaías et al.(Eds.)

PAÍS DE PUBLICACIÓN: Portugal

AÑO DE PUBLICACIÓN: 2004

ISBN: 972-98947-5-2

INDICIOS DE CALIDAD:

- El artículo tiene 1 cita en Google Scholar

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (2)

TÍTULO DEL LIBRO: Proceedings of the 6th International Conference on Web Engineering

TÍTULO DEL CAPÍTULO: An Adaptation Method by Feedback in an Evolutionary System

CLAVE: Capítulo de Libro

COLECCIÓN: ACM International Conference Proceeding Series

PÁGINAS: Desde: 161 Hasta: 168

EDITORIAL: ACM Press

CIUDAD: New York

PAÍS DE PUBLICACIÓN: Estados Unidos de América (los)

AÑO DE PUBLICACIÓN: 2006

ISBN: 1-59593-352-2

INDICIOS DE CALIDAD:

- El artículo tiene 4 citas en Google Scholar (2 no autocitas) y 2 citas en Scopus

AUTORES: Molina Ortiz, F., Medina Medina, N., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (1)

TÍTULO DEL LIBRO: Proceedings of the IADIS International Conference - Applied Computing 2006

TÍTULO DEL CAPÍTULO: A Tool for the Development of Adaptive Evolutionary Hypermedia Systems

CLAVE: Capítulo de Libro

COLECCIÓN: IADIS

PÁGINAS: Desde: 595 Hasta: 599

EDITORIAL: IADIS Press. Nuno Guimaraes et al. (Eds.)

PAÍS DE PUBLICACIÓN: Portugal

AÑO DE PUBLICACIÓN: 2006

ISBN: 972-8924-09-7

INDICIOS DE CALIDAD:

- El artículo tiene 3 citas (2 no autocitas) en Google Scholar

- El artículo fue presentado en "IADIS International Conference - Applied Computing", clasificado con ranking C en CORE

AUTORES: Molina Ortiz, F., Medina Medina, N., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (1)

TÍTULO DEL LIBRO: Workshop Proceedings of the Sixth International Conference on Web Engineering

TÍTULO DEL CAPÍTULO: An Author Tool Based on SEM-HP for the Creation and Evolution of Adaptive Hypermedia Systems

CLAVE: Capítulo de Libro

VOLUMEN: 155

COLECCIÓN: ACM International Conference Proceeding Series

PÁGINAS: Desde: 12 Hasta: 20

EDITORIAL: ACM Press

CIUDAD: New York

PAÍS DE PUBLICACIÓN: Estados Unidos de América (los)

AÑO DE PUBLICACIÓN: 2006

ISBN: 1-59593-435-9

INDICIOS DE CALIDAD:

- El artículo tiene 8 citas en Google Scholar (5 no autocitas)

AUTORES: Molina Ortiz, F., Medina Medina, N., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (1)

TÍTULO DEL LIBRO: Computer Aided Systems Theory - Extended Abstract

TÍTULO DEL CAPÍTULO: Applying a Semantic Hypermedia Model to Adaptive Concept Maps in Education

CLAVE: Capítulo de Libro

COLECCIÓN: Eurocast

PÁGINAS: Desde: 146 Hasta: 148

EDITORIAL: Alexis Quesada-Arencibia et al. (Eds.)

CIUDAD: Las Palmas de Gran Canaria

PAÍS DE PUBLICACIÓN: España

AÑO DE PUBLICACIÓN: 2007

ISBN: 978-84-690-3603-7

AUTORES: Arroyo, R., Medina Medina, N., Hornos Barranco, M. J., Molina Ortiz, F.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 4 (4)

TÍTULO DEL LIBRO: Third International Conference on Web Information Systems and Technologies - Proceedings

TÍTULO DEL CAPÍTULO: Applying Quality Hypermedia Design Principles to a Web Educational System

CLAVE: Capítulo de Libro

PÁGINAS: Desde: 458 Hasta: 463

EDITORIAL: INSTICC Press

PAÍS DE PUBLICACIÓN: Portugal

AÑO DE PUBLICACIÓN: 2007

ISBN: 978-972-8865-79-5

INDICIOS DE CALIDAD:

- El artículo fue presentado en "The Third International Conference on Web Information Systems and Technologies (WEBIST)", clasificada con ranking C en CORE

- El artículo tiene 1 cita (no autocita) en Google Scholar

6.2 Artículos en revistas

6.2.1 Publicaciones recogidas en el "Journal Citation Report"

AUTORES: Paderewski Rodríguez, P., Torres Carbonell, J. J., Rodríguez Fórtiz, Mª J., Medina Medina, N., Molina Ortiz, F.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 5 (5)

TÍTULO: A Software System Evolutionary and Adaptive Framework: Application to Agent-Based Systems

CLAVE: Artículo

NOMBRE DE LA REVISTA: Journal of Systems Architecture

VOLUMEN: 50

PÁGINAS: Desde: 407 Hasta: 416

EDITORIAL: Elsevier Science BV

PAÍS DE PUBLICACIÓN: Países Bajos (los)

AÑO DE PUBLICACIÓN: 2004

ISSN: 1383-7621

INDICIOS DE CALIDAD:

- El índice de impacto (Impact Factor) en JCR en el año 2004 es 0.242

- La revista pertenece a la categoría: Computer Science, Hardware & Architecture, y ocupa la posición 39 de 44 en la categoría Computer Science, Hardware & Architecture

- El artículo tiene 9 citas en Google Scholar (1 no autocita), 2 citas en ISI Web of Science y 3 citas en Scopus

- El artículo fue seleccionado después de ser presentado en la International Conference on Software Engineering Research and Practice, que ocupa la posición 108 de 735 en Computer Science Conference Ranking en el área Applications/Education/Software/Theory/Communications. Con índice de impacto 0,75 normalizado a 1

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (2)

TÍTULO: Diversity of Structures and Adaptive Methods on an Evolutionary Hypermedia System

CLAVE: Artículo

NOMBRE DE LA REVISTA: IEE Proceedings Software

VOLUMEN: 152

PÁGINAS: Desde: 119 Hasta: 126

EDITORIAL: IEE-INST ELEC ENG

PAÍS DE PUBLICACIÓN:

Reino Unido de Gran Bretaña e Irlanda del Norte (el)

AÑO DE PUBLICACIÓN: 2005

ISSN: 1462-5970

INDICIOS DE CALIDAD:

- El índice de impacto (Impact Factor) en JCR en el año 2005 es 0.3

- La revista pertenece a la categoría: Computer Science, Software Engineering, y ocupa la posición 309 de 352 en la categoría Computer Science

- El artículo tiene 10 citas en Google Scholar, 7 citas en Scopus y 3 citas en ISI Web of Science (1 no autocita)

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (2)

TÍTULO: Adaptation and User Modeling in Hypermedia Learning Environments using the SEM-HP Model and the JSEM-HP Tool

CLAVE: Artículo

NOMBRE DE LA REVISTA: Knowledge and Information Systems

VOLUMEN: 29 (3)

PÁGINAS: Desde: 629 Hasta: 656

EDITORIAL: Springer-Verlag

CIUDAD: London

PAÍS DE PUBLICACIÓN: Reino Unido de Gran Bretaña e Irlanda del Norte (el)

AÑO DE PUBLICACIÓN: 2010 (on-line), 2011 (impresa)

ISSN: 0219-1377

INDICIOS DE CALIDAD:

- El índice de impacto (Impact Factor) en JCR en el año 2010 es 2.0

- La revista pertenece a la categoría: Computer Science, Information System, y ocupa la posición 23 de 106 en la categoría Computer Science, Information System. Por lo tanto, la revista se encuentra en el primer cuartil de Computer Science, Information System

- El artículo tiene 28 páginas

6.2.2 Publicaciones recogidas en otras bases de datos de "ISI Web of Science"

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L., Parets Llorca, J.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 4 (2)

TÍTULO: Personalized Guided Routes in an Adaptive Evolutionary Hypermedia System

CLAVE: Artículo

NOMBRE DE LA REVISTA O MEDIO DE DIFUSIÓN: Lectures Notes in Computer Science

VOLUMEN: 2809

PÁGINAS: Desde: 196 Hasta: 207

EDITORIAL: Springer-Verlag

CIUDAD: Berlin

PAÍS DE PUBLICACIÓN: República Federal de Alemania (la)

AÑO DE PUBLICACIÓN: 2003

ISSN: 0302-9743

INDICIOS DE CALIDAD:

- El trabajo aparece indexado en DBLP, Scirus y Scopus

- El año anterior y el siguiente, la revista aparece indexada en JCR

- La revista tiene evaluación por pares y comité científico internacional

- El artículo tiene 5 citas en Google Scholar (1 no autocita), 4 citas en Scopus y 1 cita en ISI Web of Science

AUTORES: Molina Ortiz, F., Medina Medina, N., García Cabrera, L.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (1)

TÍTULO: Applying a Semantic Hypermedia Model to Adaptive Concept Maps in Education

CLAVE: Artículo

NOMBRE DE LA REVISTA O MEDIO DE DIFUSIÓN: Lecture Notes in Computer Science

VOLUMEN: 4739

PÁGINAS: Desde: 384 Hasta: 391

EDITORIAL: Springer

CIUDAD: Heidelberg

PAÍS DE PUBLICACIÓN: República Federal de Alemania (la)

AÑO DE PUBLICACIÓN: 2007

ISSN: 0302-9743

INDICIOS DE CALIDAD:

- La revista tiene evaluación por pares y comité científico internacional

6.2.3 Publicaciones recogidas en otras bases de datos

AUTORES: Molina Ortiz, F., García Cabrera, L., Medina Medina, N.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 3 (1)

TÍTULO: Applying Software Evolution Theory to Hypermedia Systems

CLAVE: Artículo

NOMBRE DE LA REVISTA O MEDIO DE DIFUSIÓN: Int. J. Web Engineering and Technology (IJWET)

VOLUMEN: 5

PÁGINAS: Desde: 69 Hasta: 87

EDITORIAL: Inderscience Enterprises Ltd.

CIUDAD: Genéve

PAÍS DE PUBLICACIÓN: Suiza

AÑO DE PUBLICACIÓN: 2009

ISSN: 1476-1289

INDICIOS DE CALIDAD:

- La revista aparece indexada en: Scopus, Scirus, Google Scholar, ERGS, DBLP, COMPEN, COMDB, CIS, BUSASAP y ACM

- La revista tiene evaluación externa por pares y comité científico internacional

- El artículo tiene 1 cita (no autocita) en Google Scholar

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L., Rodríguez Fórtiz, Mª J.

Nº DE AUTORES Y POSICIÓN QUE OCUPA EL DOCTORANDO ENTRE ELLOS: 4 (2)

TÍTULO: Coevolution of Models of an Adaptive Hypermedia System

CLAVE: Artículo

NOMBRE DE LA REVISTA O MEDIO DE DIFUSIÓN: IDPT'03 © SDPS

VOLUMEN: 2

PÁGINAS: Desde: 38 Hasta: 45

EDITORIAL: Society for Design and Process Science

PAÍS DE PUBLICACIÓN: Estados Unidos de América (los)

AÑO DE PUBLICACIÓN: 2003

ISSN: 1090-9389

INDICIOS DE CALIDAD:

- La revista tiene evaluación por pares y comité científico internacional

- El artículo tiene 2 citas (1 no autocita) en Google Scholar

6.3 Actas de congresos

AUTORES: Medina Medina, N., Molina Ortiz, F., García Cabrera, L., Rodríguez Fórtiz, Mª J.

TÍTULO: SEM-HP: Un Modelo para el Desarrollo de Sistemas Hipermedia Adaptativos

ENTIDAD ORGANIZADORA: Universidad de Granada

CONGRESO: II Workshop de Investigación sobre Nuevos Paradigmas de Interacción en Entornos Colaborativos Aplicados a la Gestión y Difusión del Patrimonio Cultural (COLINE'02)

TIPO DE PARTICIPACIÓN: Ponencia

LUGAR DE CELEBRACIÓN: Granada, España

FECHA DE CELEBRACIÓN: FECHA INICIO: 11/11/2002 FECHA FIN: 12/11/2002

AUTORES: Medina Medina, N., García Cabrera, L., Molina Ortiz, F.

TÍTULO: Diversidad de Tipos de Navegación en un SHA

ENTIDAD ORGANIZADORA: Grupo de Investigación GEDES (Universidad de Granada)

CONGRESO: VIII Jornadas de Ingeniería del Software y Bases de Datos. Taller en Sistemas Hipermedia Colaborativos y Adaptativos (2ª edición)

TIPO DE PARTICIPACIÓN: Ponencia

PÁGINAS: Desde: 1 Hasta: 7

LUGAR DE CELEBRACIÓN: Alicante, España

FECHA DE CELEBRACIÓN: FECHA INICIO: 12/11/2003 FECHA FIN: 14/11/2003

AUTORES: Molina Ortiz, F., Medina Medina, N., García Cabrera, L.

TÍTULO: JSEM-HP: Una Herramienta de Autor para el Desarrollo de Sistemas Hipermedia Adaptativos Evolutivos

ENTIDAD ORGANIZADORA: Grupo de Investigación GEDES (Universidad de Granada)

CONGRESO: VIII Jornadas de Ingeniería del Software y Bases de Datos. IV Jornadas de Trabajo DOLMEN

TIPO DE PARTICIPACIÓN: Ponencia

PÁGINAS: Desde: 110 Hasta: 115

LUGAR DE CELEBRACIÓN: Alicante, España

FECHA DE CELEBRACIÓN: FECHA INICIO: 12/11/2003 FECHA FIN: 14/11/2003

AUTORES: Medina Medina, N., García Cabrera, L., Molina Ortiz, F.

TÍTULO: Punto de Partida para AMENITIES: Un Modelo Hipermedia Evolutivo y Adaptativo

ENTIDAD ORGANIZADORA: Universidad de Albacete

CONGRESO: I Jornadas de Trabajo ADACO

TIPO DE PARTICIPACIÓN: Ponencia

LUGAR DE CELEBRACIÓN: Albacete, España

FECHA DE CELEBRACIÓN: FECHA INICIO: 10/02/2005 FECHA FIN: 10/02/2005

AUTORES: Molina Ortiz, F., Medina Medina, N.

TÍTULO: Herramienta de Desarrollo de Sistemas Hipermedia Adaptativos

ENTIDAD ORGANIZADORA: Fundación Española para la Ciencia y la Tecnología (FECYT) y Junta de Andalucía

CONGRESO: Semana de la Ciencia y la Tecnología

TIPO DE PARTICIPACIÓN: Ponencia

PÁGINAS: Desde: 60 Hasta: 60

LUGAR DE CELEBRACIÓN: Granada, España

FECHA DE CELEBRACIÓN: FECHA INICIO: 07/11/2005 FECHA FIN: 13/11/2005

AUTORES: N., Medina Medina, N., Padilla Zea, Cabrera Cuevas, M., Molina Ortiz, F., García Cabrera, L.

TÍTULO: A Model Based on Semantic Nets to Support Evolutionary and Adaptive Hypermedia Systems

ENTIDAD ORGANIZADORA: Alexandra Cristea & Rosa Carro (at EC-TEL 2009)

CONGRESO: A3H: Seventh International Workshop on Authoring of Adaptive and Adaptable Hypermedia

TIPO DE PARTICIPACIÓN: Ponencia

LUGAR DE CELEBRACIÓN: Nice, Francia

FECHA DE CELEBRACIÓN: FECHA INICIO: 29/10/2009 FECHA FIN: 29/10/2009

REFERENCIAS

(Berners-Lee et al., 2002) Berners-Lee, T., Hendler, J. and Lassila, O. The semantic Web. Scientific American Special Online Issue, 24–30, 2002.

(Bieber et al., 1997) Bieber, M., Vitali, F., Ashman, H., Balasubramanian, V. and Oinas-Kukkonen, H. Fourth generation hypermedia: some missing links for the World Wide Web. International Journal of Human-Computer Studies, 47(1), 31–65, 1997.

(Brusilovsky, 1996) Brusilovsky, P. Methods and techniques of adaptive hypermedia', User Modeling and User-Adapted Interaction, 6, 87–129, 1996.

(Bush et al., 1945) Bush, V. As we may think. The Atlantic Month, 176, 101–108, 1945.

(Cañas et al., 2004) Cañas, A. J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., Gómez, G., Arroyo, M. and Carvajal, R. CmapTools: a knowledge modeling and sharing environment. Concept Maps: theory, methodology, technology', First International Conference on Concept Mapping, 125–133, 2004.

(Ceri et al., 2002) Ceri, S., Matera, M., Bongio, A., Fraternali, P. and Comai, S. Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers, 2002.

(Colmenero, 2005) Colmenero-Ruiz, M. J. Introducción al modelo Topic Maps (ISO/IEC13250:2003). Revista Digital de Biblioteconomía e Ciéncia da Informaçao, 3 (1), 77-102, 2005.

(De Bra et al., 1998) De Bra, P., Houben, G.J. and Wu, H. AHAM: a reference model to support adaptive hypermedia authoring. InfWet'98 Conference, http://wwwis.win.tue.nl/~debra/infwet98/ paper.pdf. 1998.

(De Bra et al., 2003) De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D. and Stash, N. AHA! The adaptive hypermedia architecture.ACM Hypertext Conference, Nottingham, UK. 2003.

(De Bra et al., 2004) De Bra, P., Aroyo, L. and Chepegin, I. (2004) The next big thing: adaptive web-based systems. J. Digit. Inf., 5 (1), 2004.

(Díaz et al., 2005) Díaz, P., Montero, S. and Aedo, I. Ingeniería de la Web y patrones de diseño. Prentice Hall, 2005.

(García, 2001) García-Cabrera, L. SEM-HP: A modelo semántico, sistémico y evolutivo para el desarrollo de sistemas hipermedia. Tesis doctoral, 2001.

(García et al., 2002) García-Cabrera, L., Rodríguez, M.J. and Parets, J. Evolving hypermedia systems: a layered software architecture. Journal of Software Maintenance and Evolution: Research and Practice, John Wiley & Sons, Ltd., 14 (5), 389–405, 2002.

(Halasz y Schwartz, 1990) Halasz, F. and Schwartz, M. The Dexter hypertext reference model. NIST Hypertext Standardization Workshop, 95–133, 1990.

(Holt et al., 2000) Holt, R. C., Andy Schrr, Susan Elliott Sim, and Andreas Winter. Gxl: A graph-based standard exchange format for reengineering. Univ., Inst. für Informatik, 2000.

(Hurtado Torres, 2002) Hurtado-Torres, M.V. Un Modelo de Integración Evolutivo entre Sistemas de Información y Sistemas de Ayuda a la Decisión. Tesis Doctoral, 2002.

(JGraph) JGraph. http://www.jgraph.com.

(Kibby et al., 1990) Kibby, M., Mayes, T. and Anderson, T. Learning about learning from hypertext. Designing Hypermedia for Learning, 227–250, 1990.

(Koch y Wirsing, 2002) Koch, N. and Wirsing, M. The Munich reference model for adaptive hypermedia applications. 2nd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, 213–222, 2002.

(Lowe y Hall, 1999) Lowe, D. and Hall, W. Hypermedia and the Web: An Engineering Approach, Wiley, 1999.

(Medina et al., 2002) Medina, N., García, L., Torres, J.J. and Parets, J. Evolution in adaptative hypermedia systems. First International Workshop on Principles of Software Evolution, 34–38, 2002.

(Medina, 2004) Medina-Medina, N. Un modelo de adaptación integral y evolutivo para sistemas hipermedia. El subsistema de aprendizaje de SEM-HP. Tesis doctoral, 2004.

(Medina et al., 2005) Medina-Medina, N., Molina-Ortiz, F. and García-Cabrera, L. Diversity of structures and adaptive methods on an evolutionary hypermedia system. Journal IEE Proc.-Software, 152 (3), 119–126, 2005.

(Medina et al., 2006) Medina-Medina, N., Molina-Ortiz, F. and García-Cabrera, L. An adaptation meted by feedback in an evolutionary hypermedia system. Sixth International Conference on Web Engineering, 161–168, 2006.

(Medina et al., 2011) Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L. Adaptation and user modeling in hypermedia learning environments using the SEM-HP model and the JSEM-HP tool. Knowledge and Information Systems, 29 (3), 629–656, 2011 (DOI: 10.1007/s10115-010-0357-1).

(Molina et al., 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, 77-83, 2002.

(Molina et al., 2006) Molina-Ortiz, F., Medina-Medina, N. and García-Cabrera, L. An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems. Sixth International Conference on Web Engineering, 155, 2006.

(Nilsson y Palmr, 1999) Nilsson, M. and Matthias Palmr. Conzilla: towards a concept browser. Technical report, NADA (Numerical Analysis and Computing Science), KTH (Royal Institute of Technology), Sweden, 1999.

(Parets, 1995) 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 y Torres, 1996) Parets, J. and Torres, J.C. Software maintenance versus software evolution: an approach to software systems evolution. IEEE Symposium and Workshop on Engineering of Computer-Based Systems, 34–141, 1996.

(Rivlin et al., 1994) Rivlin, E., Botafogo, R. A. and Shneiderman, B. Navigation in hyperspace: designing a structure-based toolbox. Communications of the ACM (CACM), 37 (2), 87–96, 1994.

(Schwabe et al., 1996) Schwabe, D., Rossi, G. and Barbosa, S. Systematic hypermedia application design with OOHDM. ACM International Conference on Hypertext, 116-128, 1996.

(Stotts y Furuta, 1989) Stotts, P. D. and Furuta, R. Programmable browsing semantics in Trellis. Hypertext'89 Conference, 27–42. 1989.

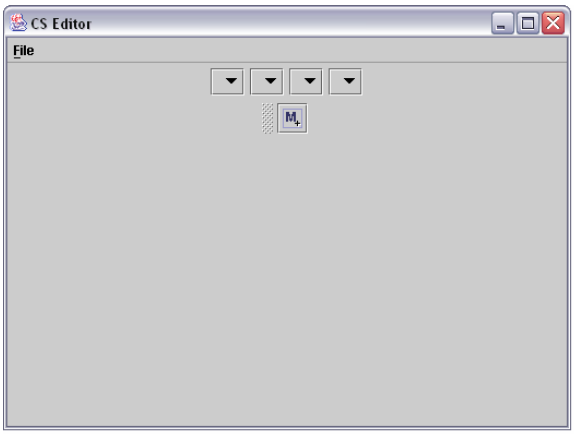
(Tamine-Lechani et al., 2009) Tamine-Lechani, L., Boughanem, M. and Daoud, M. Evaluation of contextual information retrieval effectiveness: overview of issues and research. J Knowledge and Information System. doi:10.1007/s10115-009-0231-1

(Wang y Rada, 1998) Wang, W. and Rada, R. Structured hypertext with domain semantics, ACM Trans. Information Systems, 16 (4), 1998.

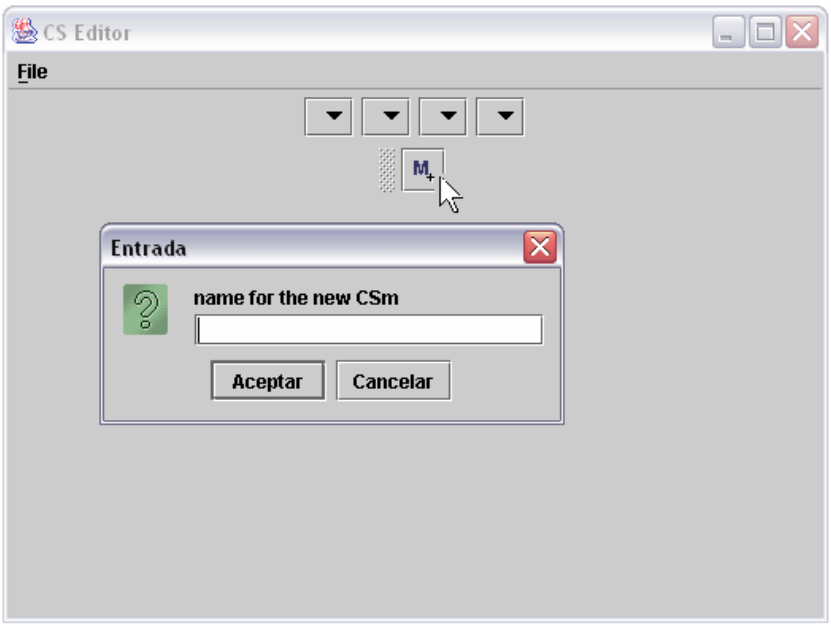
APÉNDICE 1: MANUAL DE USUARIO (DESARROLLO DEL SISTEMA)

Fase de memorización

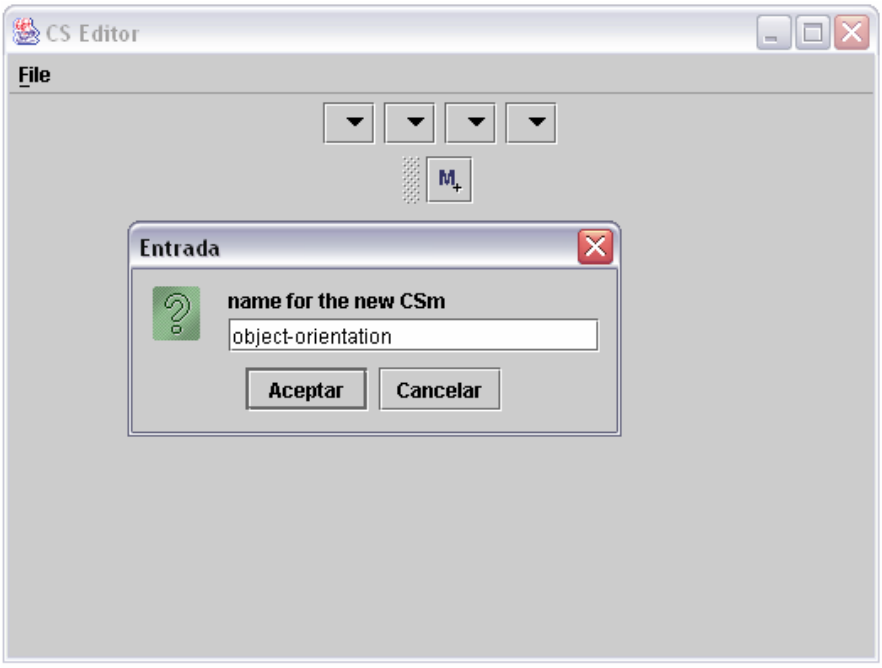
Paso I: Para empezar a trabajar (Medina, 2004) abrir el editor de estructuras conceptuales: CS Editor.



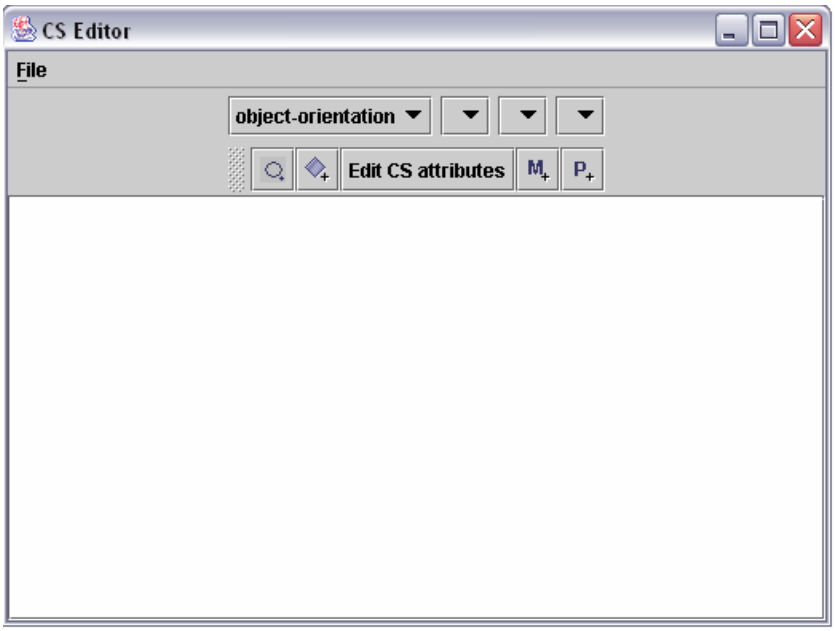
Paso II: Pulsar el botón *M+* para crear una nueva estructura conceptual de memorización.



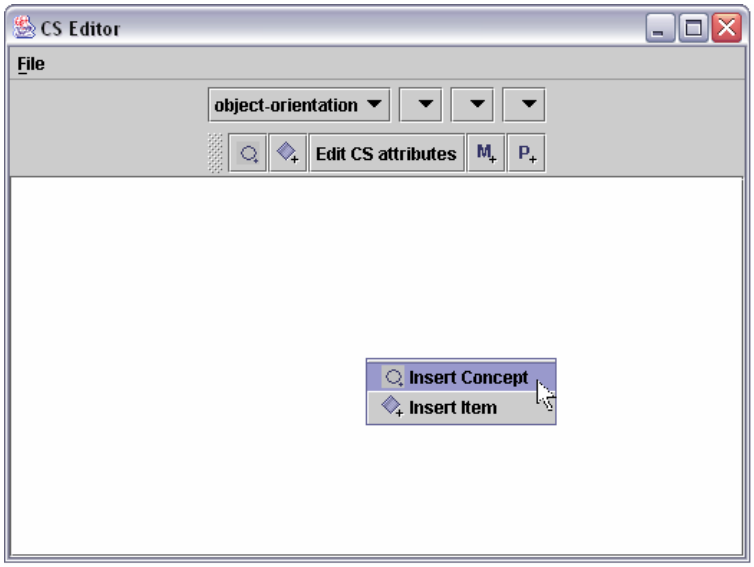
Paso III: Escribir el nombre de la nueva estructura conceptual de memorización, *object-orientation* en el ejemplo.



Paso IV: Empezar a trabajar con la estructura conceptual de memorización.

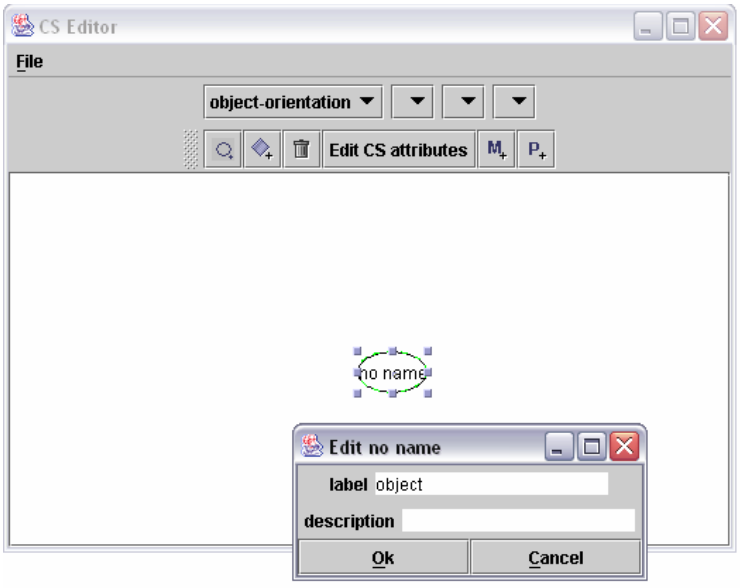


Paso V: Para crear un nuevo concepto pulsar el botón derecho del ratón sobre la posición donde se desea insertarlo, y seleccionar la opción *Insert Concept* en el menú que aparece.

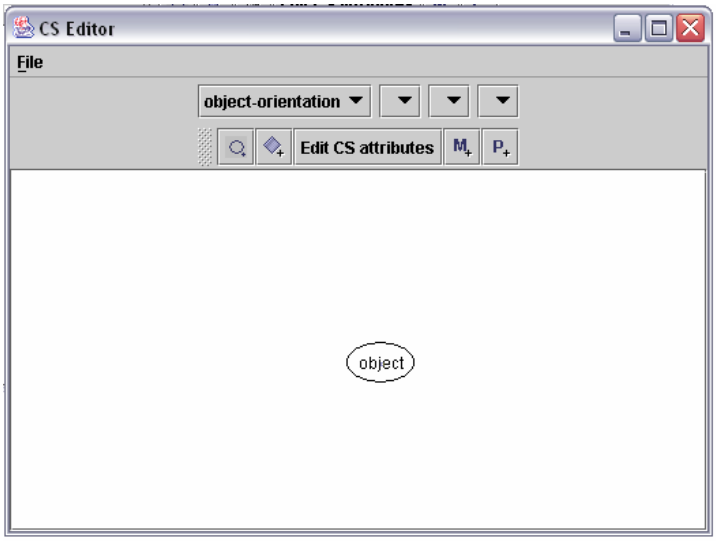


También se puede usar directamente el botón  del menú superior. En tal caso, el concepto se inserta en la esquina superior izquierda. Luego se puede colocar en la posición que se desee seleccionando y arrastrando con el ratón.

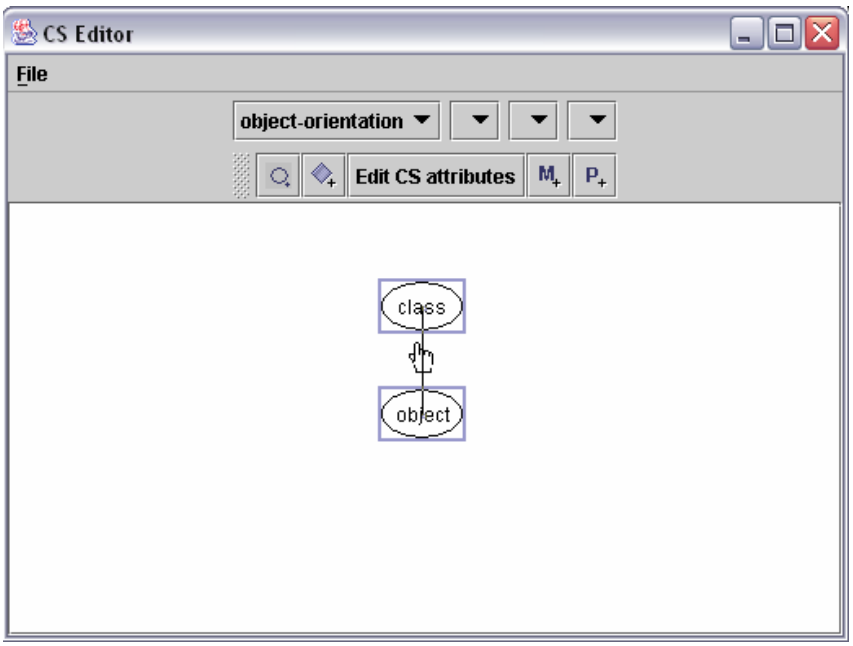
Paso VI: Escribir el nombre del concepto en el cuadro que aparece para ello. Si se desea, también se puede escribir una descripción del concepto.



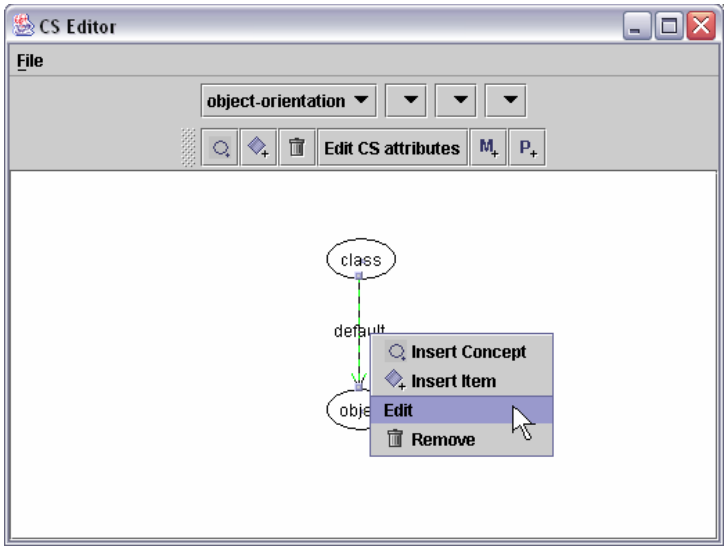
Paso VII: Seguir creando conceptos. Si se intenta introducir otro concepto con el mismo nombre que uno ya existente el sistema genera un mensaje de error y rechaza la acción.



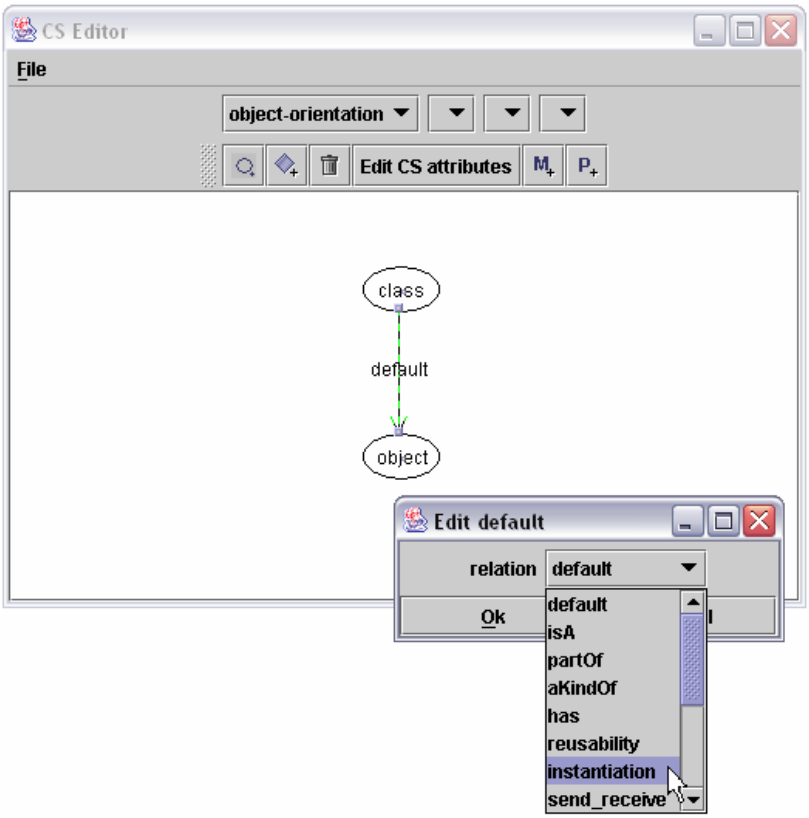
Paso VIII: Establecer una relación conceptual entre dos conceptos, situando el cursor en el centro del concepto origen y arrastrando hasta el centro del concepto destino.



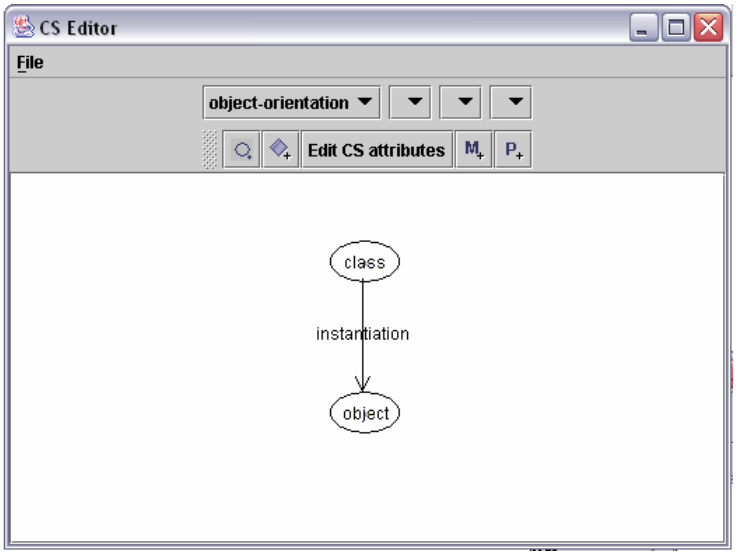
Paso IX: Pulsar el botón derecho del ratón sobre la relación conceptual para editar (*Edit*) sus propiedades.



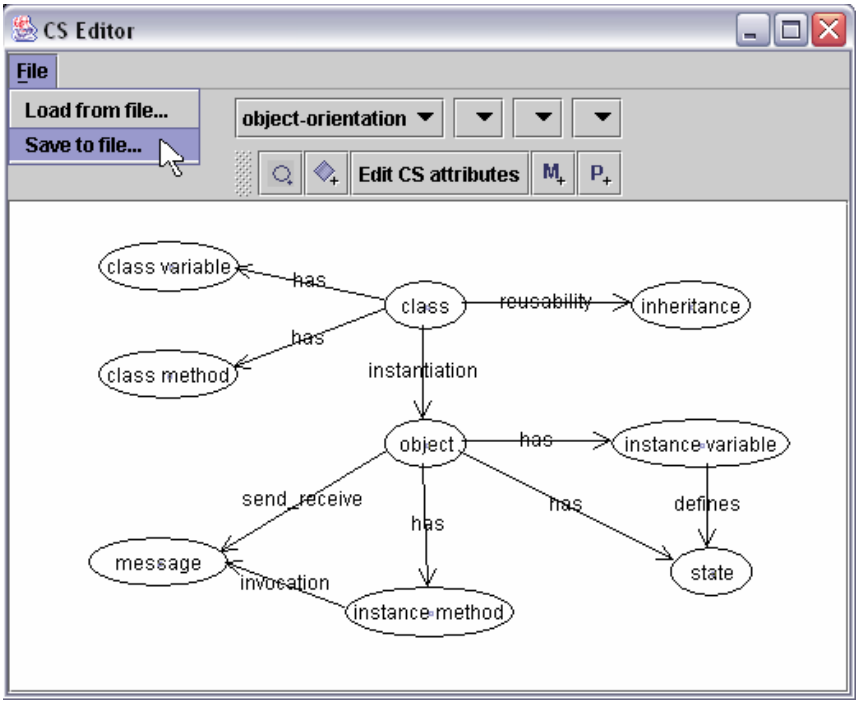
Paso X: Elegir el tipo de relación conceptual en la lista desplegable que aparece.



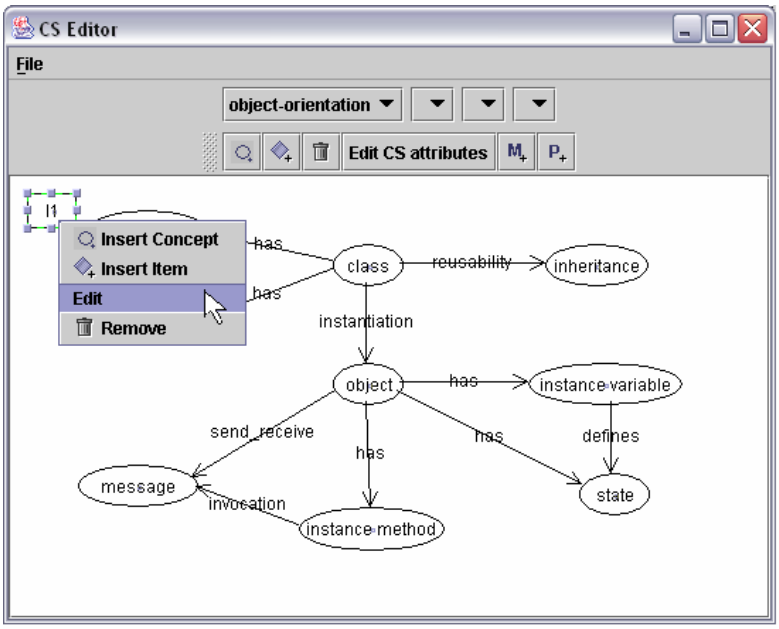
Paso XI: Crear tantos conceptos y relaciones conceptuales como se desee.



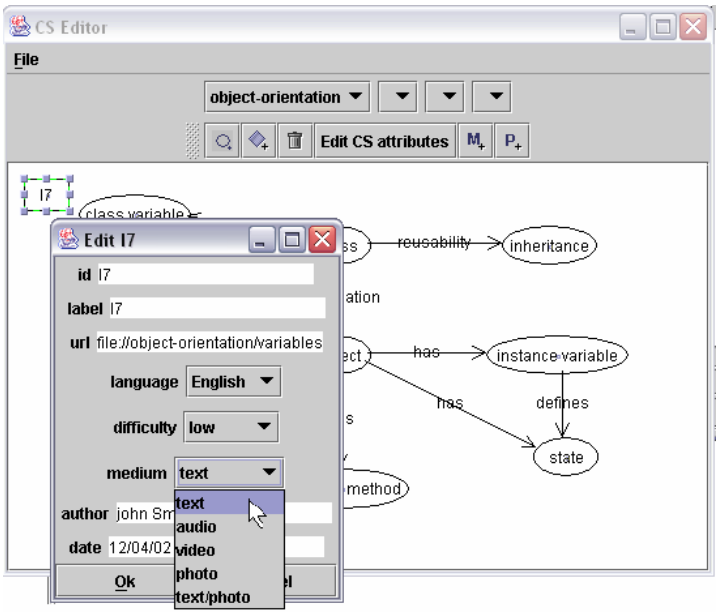
Paso XII: Pulsar *File* → *Save to file...* para guardar el trabajo realizado hasta el momento.



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.

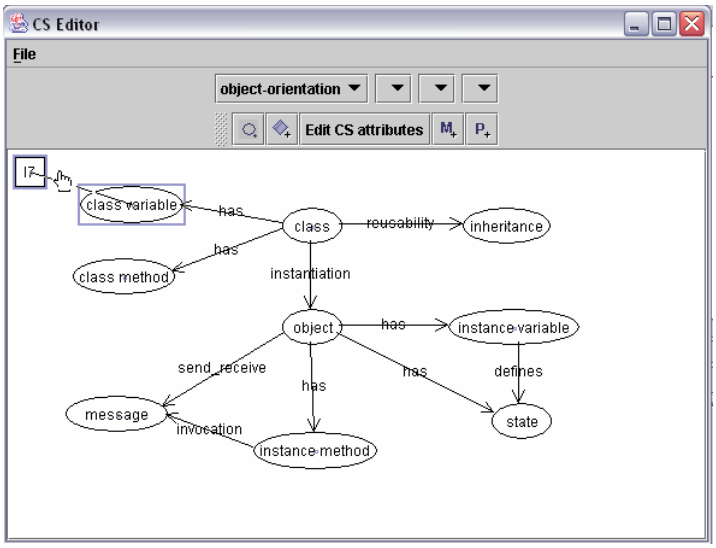


Paso XIV: Rellenar las propiedades del ítem en el cuadro de edición, indicando identificador, etiqueta, *url*, idioma, nivel de dificultad, medio, autor y fecha. Para el idioma, la dificultad y el medio, se elige un valor de una lista.

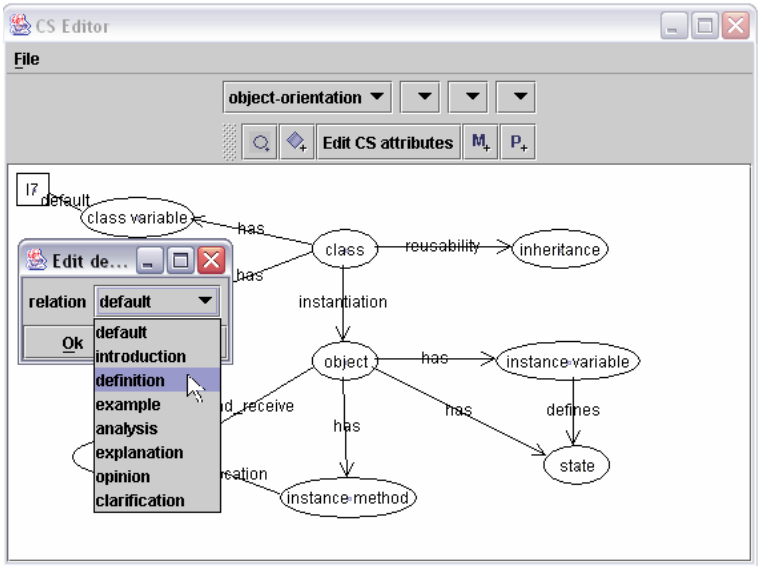


El sistema asigna por defecto el identificador del ítem siguiendo una numeración secuencial. Sin embargo, es posible modificar el identificador de un ítem, siempre que el valor dado no coincida con uno ya usado en la estructura actual.

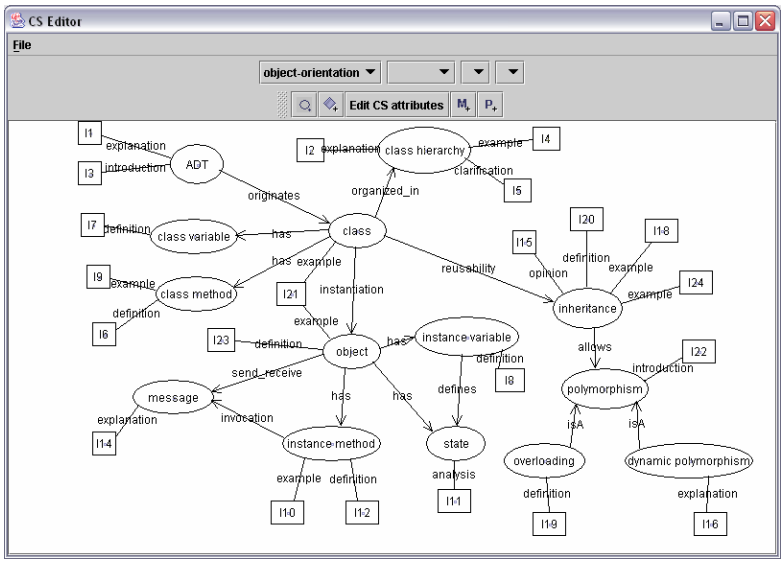
Paso XV: Crear una asociación funcional entre un ítem y un concepto, pinchando en el centro del primero y arrastrando hasta el centro del segundo.



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.

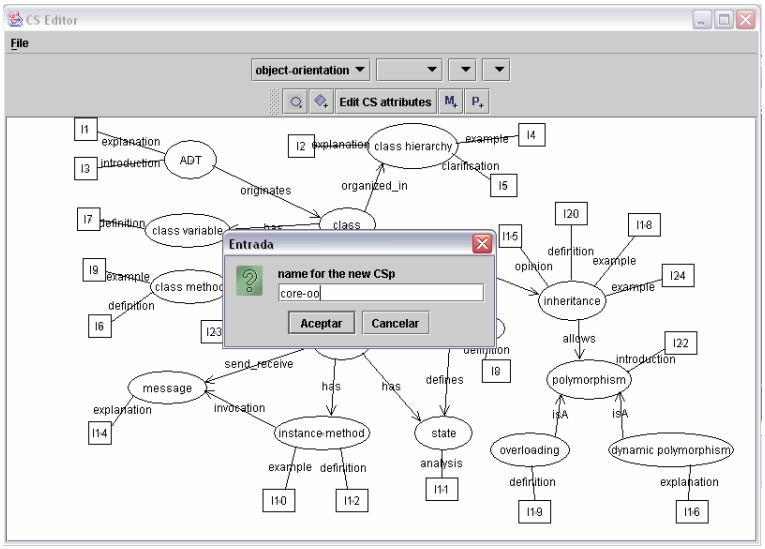


Paso XVII: Seguir insertando conceptos, ítems, relaciones conceptuales y asociaciones funcionales hasta completar la estructura conceptual de memorización final.

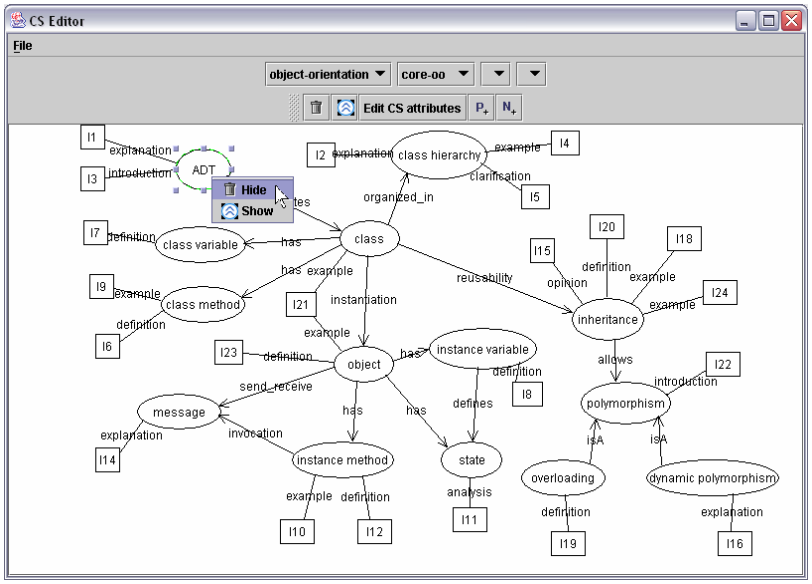


Fase de presentación

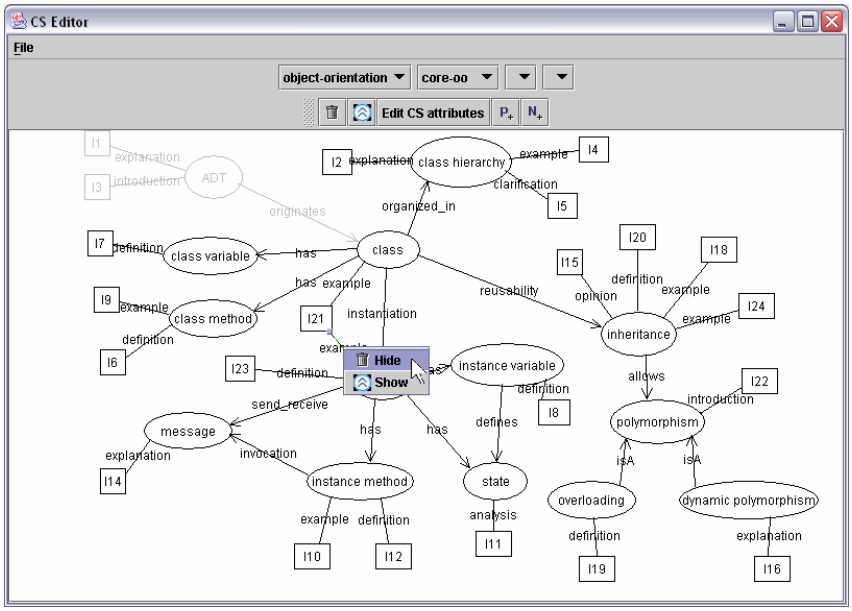
Paso I: Teniendo una estructura conceptual de memorización abierta, pulsar el botón *P+* para crear una presentación a partir de ella. Escribir el nombre de la nueva presentación en el cuadro que aparece.



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  .

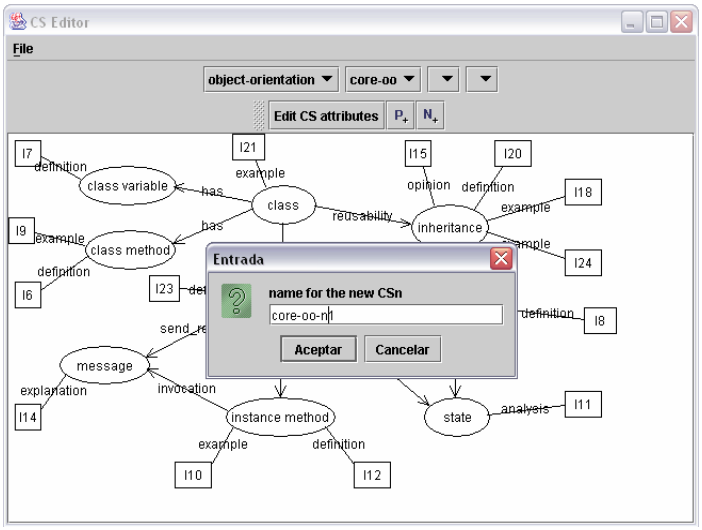


Paso III: Seguir ocultando conceptos, ítems y relaciones hasta obtener la presentación deseada.

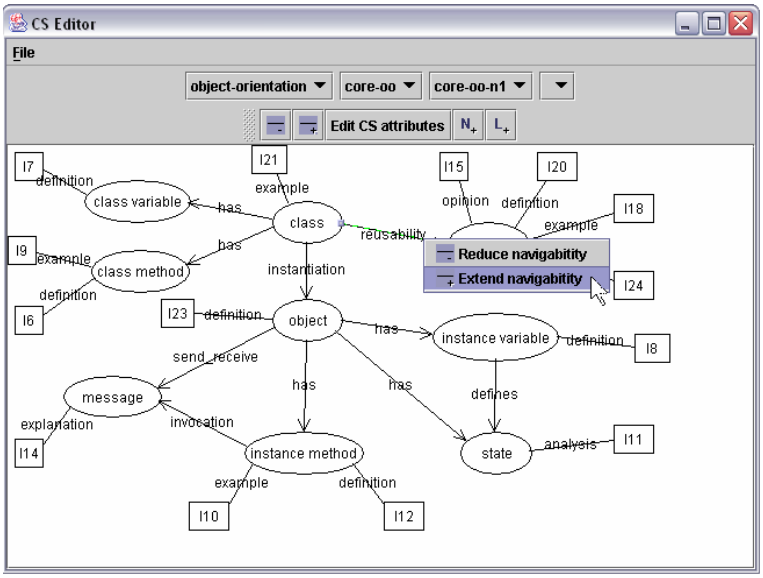


Fase de navegación

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.

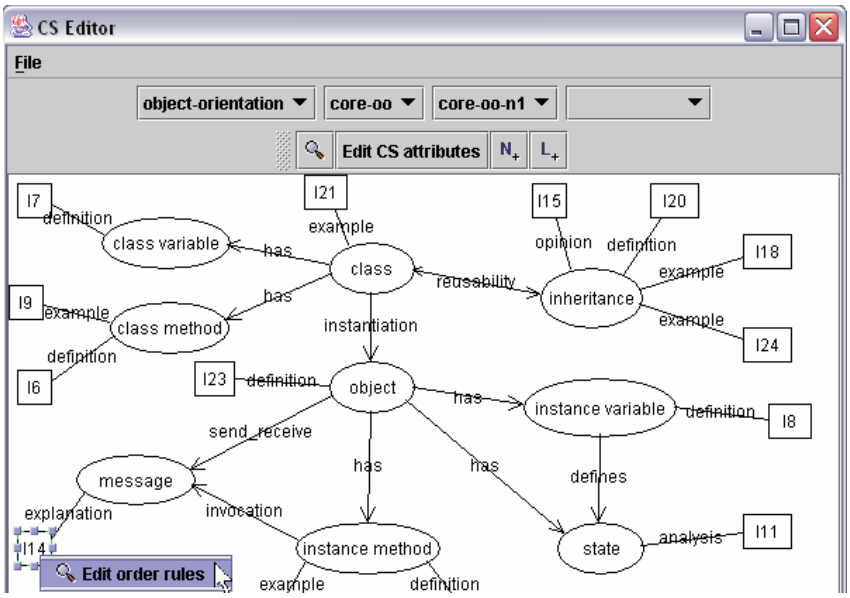


Paso II: Ampliar la navegabilidad de las relaciones conceptuales que se desee, pulsando el botón derecho sobre la relación y seleccionando *Extend navigability*. Después se puede reducir con la opción *Reduce navigability*. Las relaciones con navegabilidad extendida se mostrarán con doble flecha.

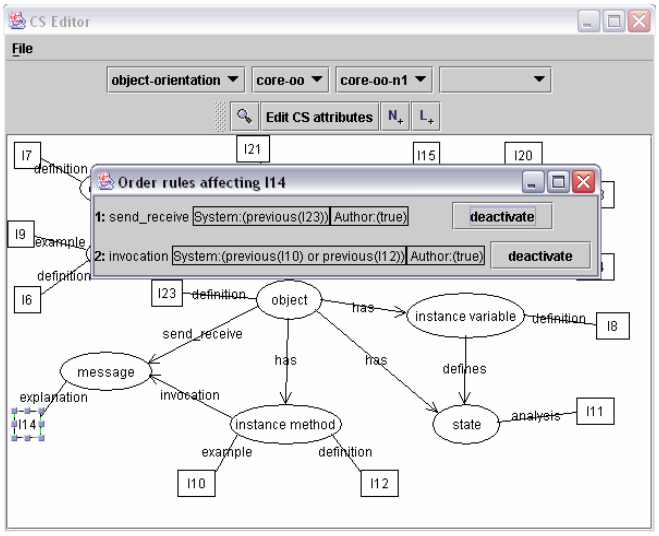


También se pueden usar los botones  que aparecen en el menú superior.

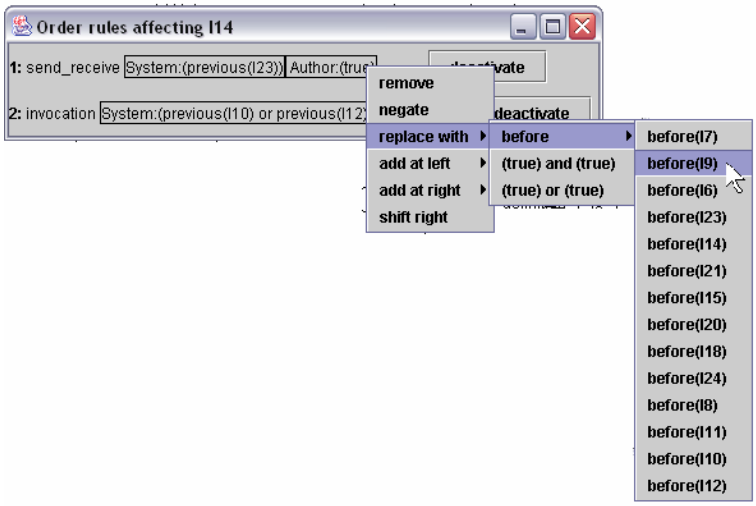
Paso III: Obtener las reglas de orden creadas por defecto para un ítem, pulsando botón derecho sobre éste y seleccionando la opción de edición (*Edit order rules*).



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.

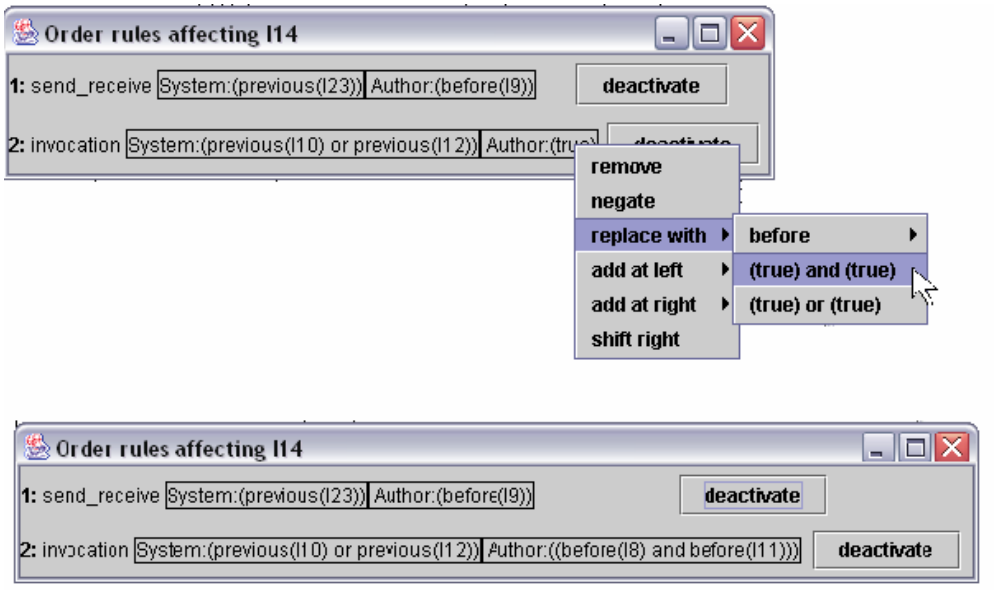


Paso V: Reemplazar el predicado *true* de la fórmula de autor de la primera regla por el predicado *before(I9)*. De esta forma se exige haber visitado I9 antes de visitar I14 a través de la relación conceptual *send_receive*.



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)*.



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.

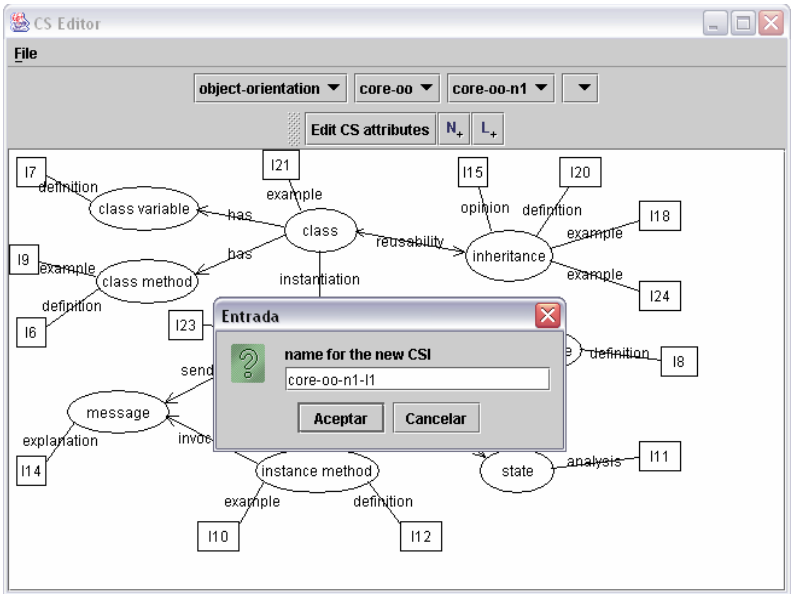
También se puede negar un predicado con la opción *negate*, añadir un nuevo predicado a la izquierda (*add at left*) o a la derecha (*add at right*) del predicado actual, o reordenar los predicados usando *shift right*.

Observe en la figura anterior 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.

Fase de aprendizaje

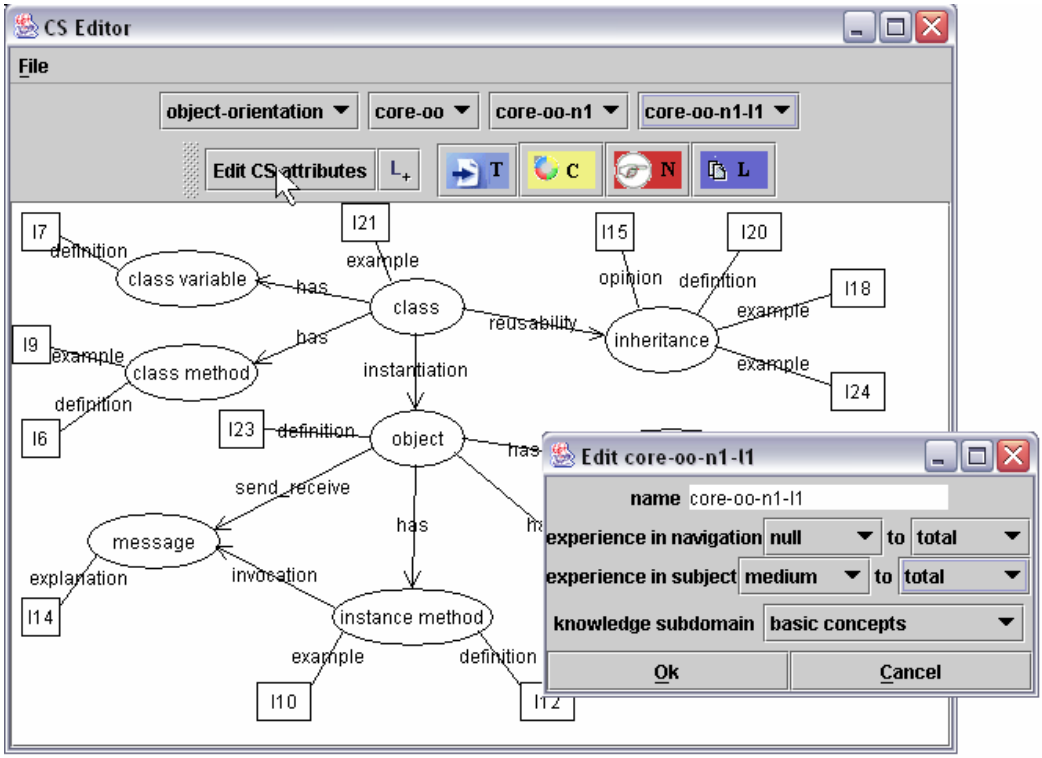
Paso I: Teniendo una estructura conceptual de navegación abierta pulsar el botón *L+* para definir una estructura conceptual de aprendizaje a partir de ella. Escribir el nombre de la nueva EC_L en el cuadro que aparece con tal fin, *core-oo-n1-l1* en el ejemplo.



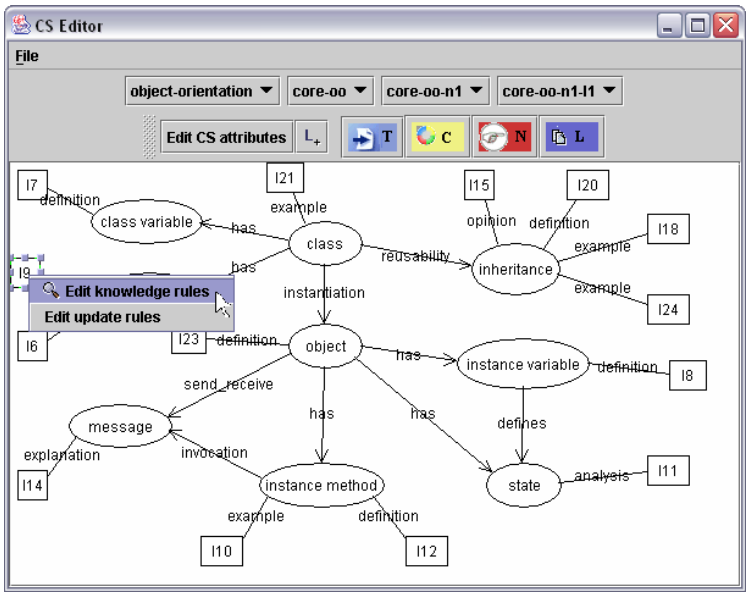
Observe que en el menú superior aparecen cuatro listas desplegables, que corresponden a cada una de las cuatro fases de diseño, y que permiten elegir:

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).

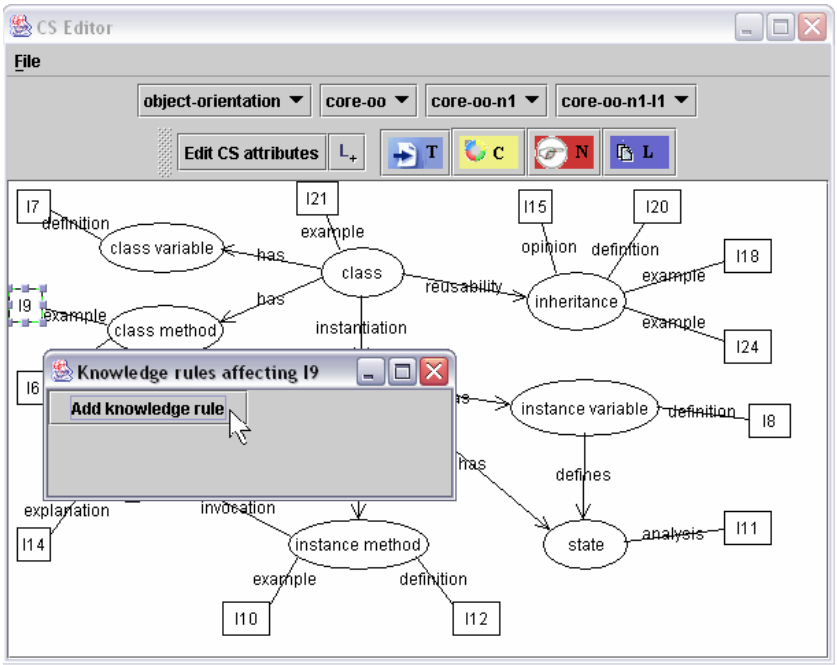
Paso II: Pulsar el botón *Edit CS attributes* para etiquetar la estructura de aprendizaje: rangos de experiencia (de navegación y en la materia) y subdominio de conocimiento.



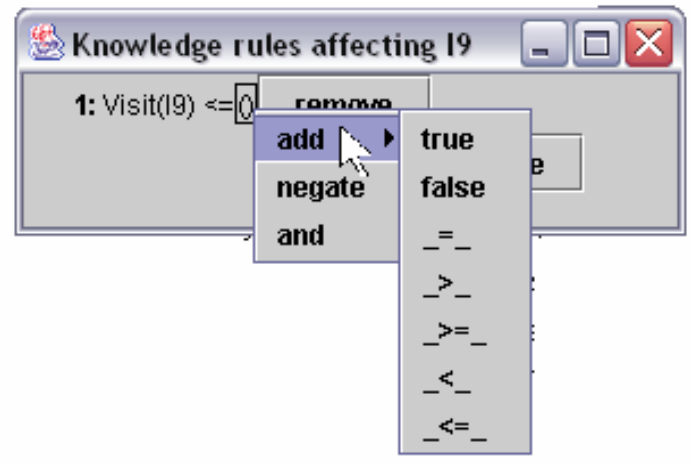
Paso III: Definir reglas de conocimiento para un ítem pulsando el botón derecho sobre éste y seleccionando la opción *Edit kowledge rules*.



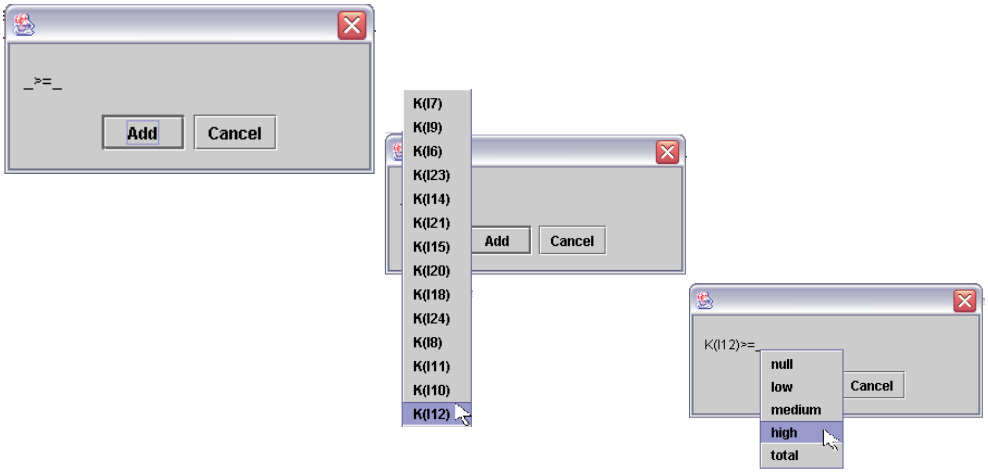
Paso IV: Por defecto no existen reglas de conocimiento asociadas a un ítem, para añadir una nueva regla pulsar el botón *Add knowledge rule*.



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.

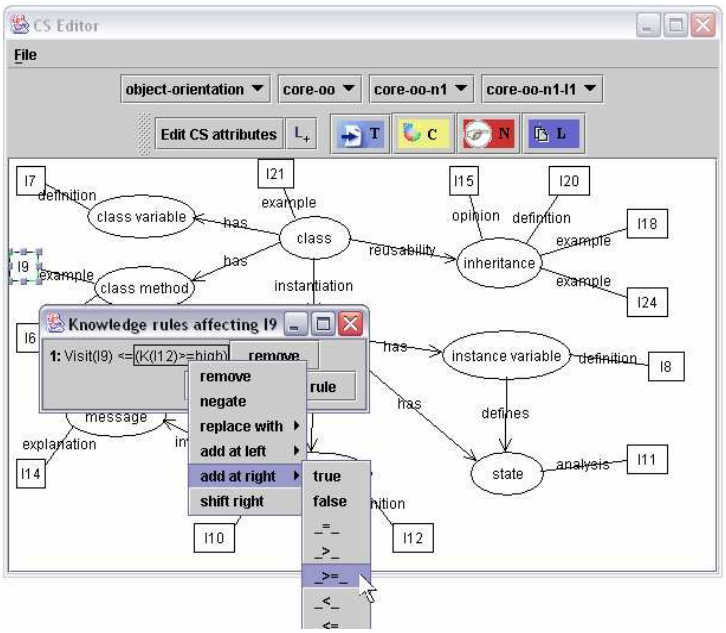


Paso VI: Aparece un cuadro para dar valor al predicado. Para establecer el término izquierdo y derecho del predicado basta elegir una opción de la lista que aparece al situarnos encima y pulsar el botón derecho. Al terminar pulsar *Add* en el cuadro inicial.

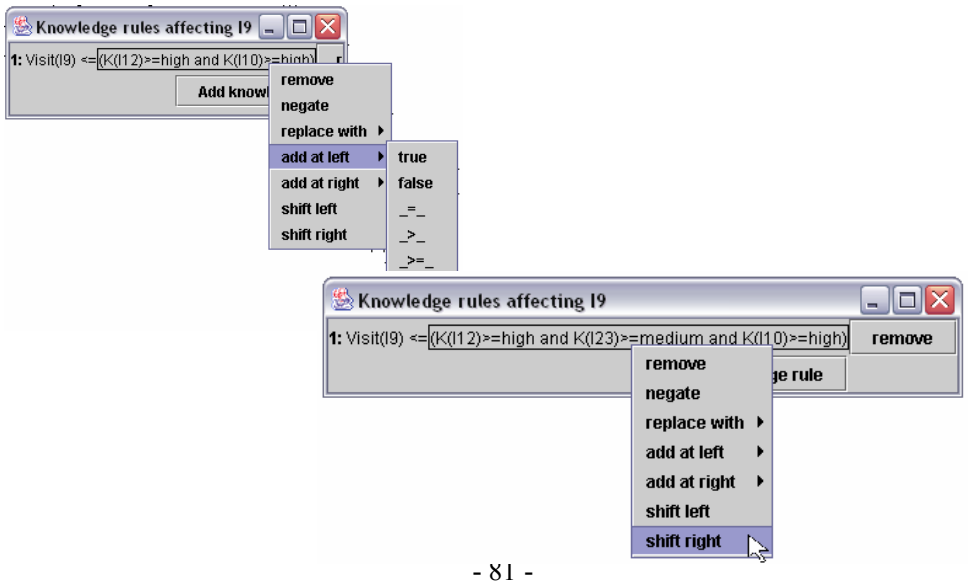


Una vez creado el predicado, se puede modificar cualquiera de sus términos pulsando el botón derecho sobre éste y eligiendo un nuevo valor de la lista. También se puede cambiar el tipo del predicado pulsando botón derecho sobre él y eligiendo la opción *replace with* con el predicado deseado.

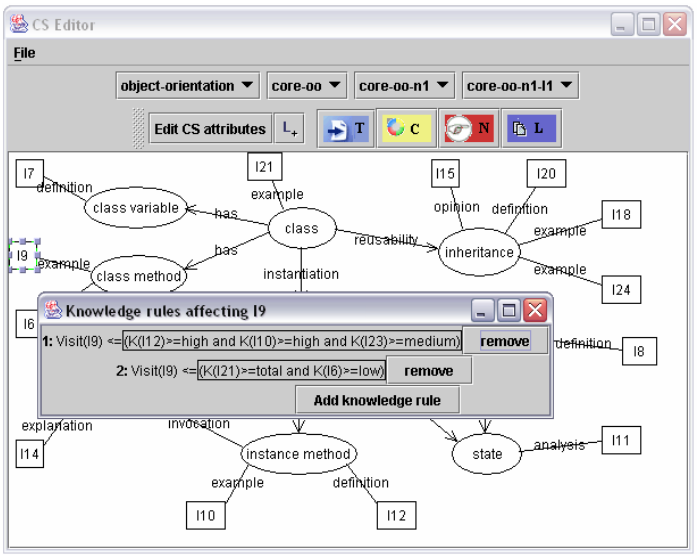
Paso VII: Añadir más prerrequisitos de conocimiento en el cuerpo de la regla. Para ello, situarse sobre el predicado de conocimiento y pulsar botón derecho. Después elegir la opción *add at left* o *add at right* según se desee añadir el nuevo predicado a la izquierda o a la derecha del predicado actual.



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).

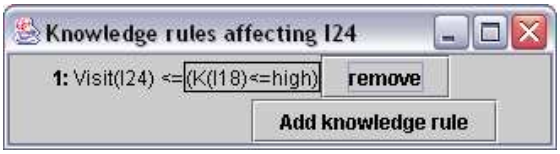


Paso IX: Para definir más reglas de conocimiento asociadas al mismo ítem pulsar *Add Knowledge rule* y crear la nueva regla como se ha descrito en los pasos anteriores.

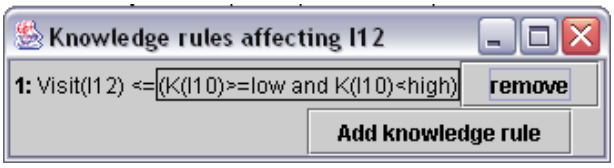


Para eliminar un predicado en el cuerpo de una regla se pulsa el botón derecho sobre él y se selecciona la opción *remove* en el menú que aparece. Para borrar toda la regla de conocimiento, pulsar el botón *remove* que aparece junto a ésta.

Paso X: Definir reglas de conocimiento para todos aquellos ítems que se desee. Por ejemplo, las que se muestran en las dos figuras siguientes.

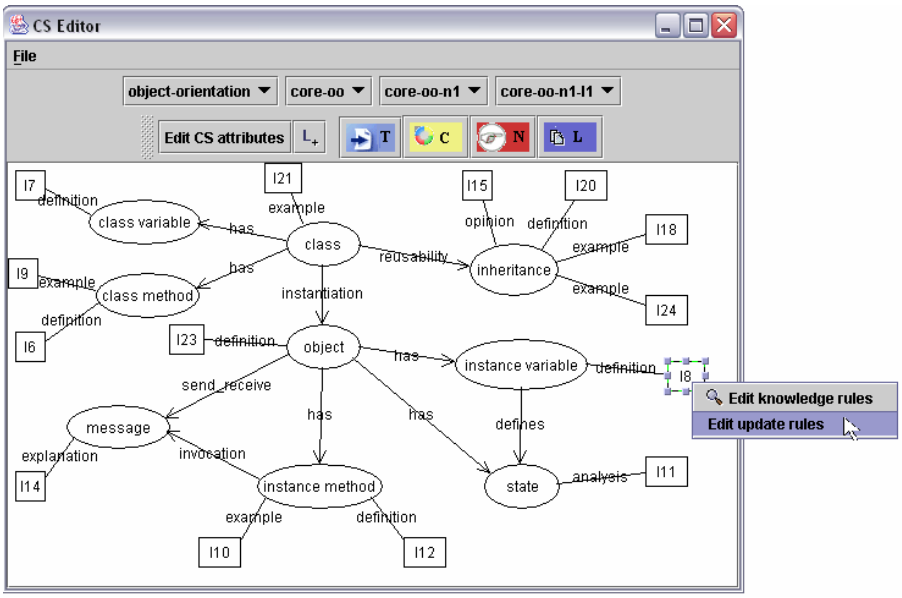


La regla de I24 sólo impone un requisito de idoneidad para su visita: que el conocimiento sobre I18 no sea total.

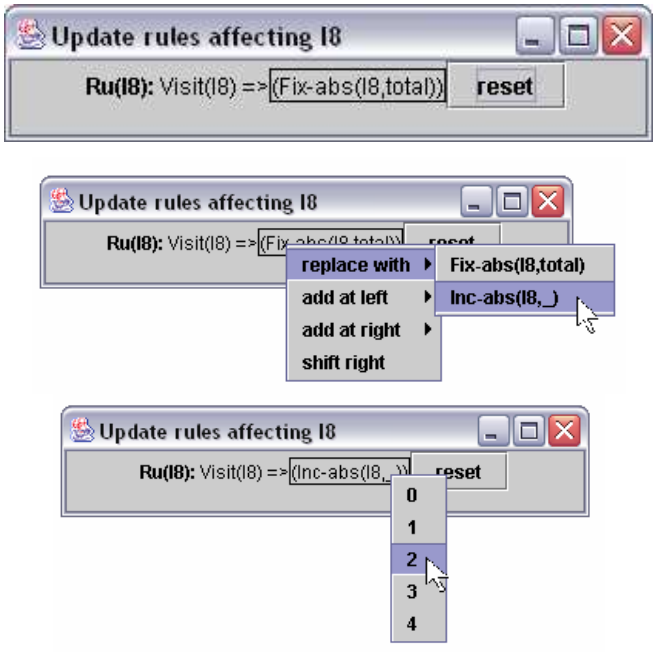


La regla de I12 impone un requisito de accesibilidad y otro de idoneidad, ambos definidos sobre el ítem I10.

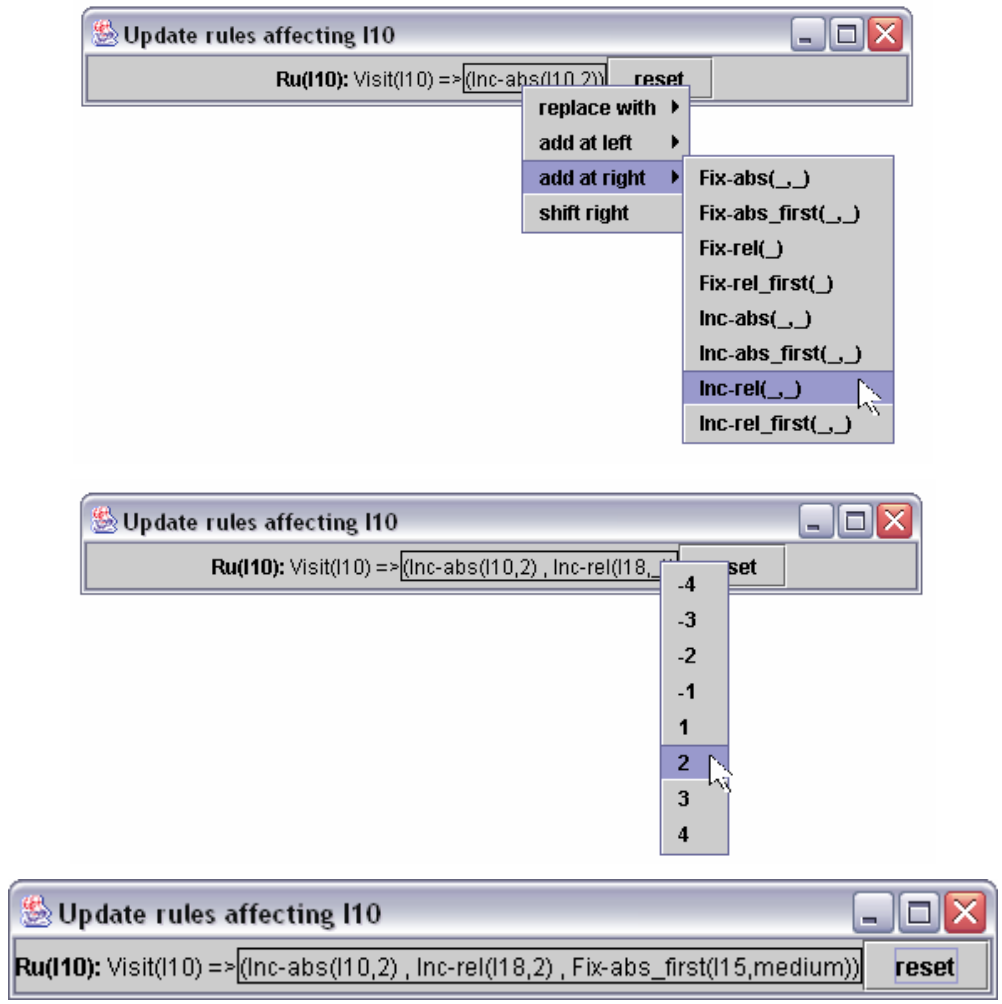
Paso XI: Editar la regla de actualización creada por defecto para un ítem, pulsando botón derecho sobre él y seleccionando la opción *Edit update rule*.



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(_, _)*.



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).



En cualquier momento se puede utilizar el botón *reset* para cambiar la regla actual por la regla de actualización generada por defecto.

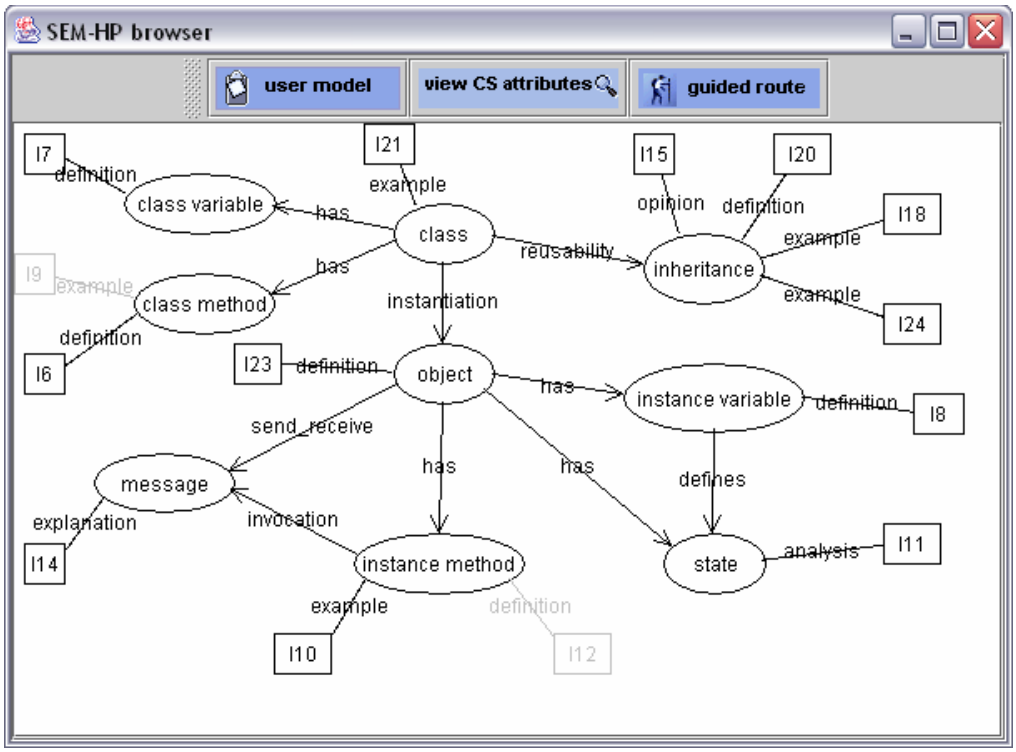
APÉNDICE 2: MANUAL DE USUARIO (NAVEGACIÓN DEL SISTEMA)

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.

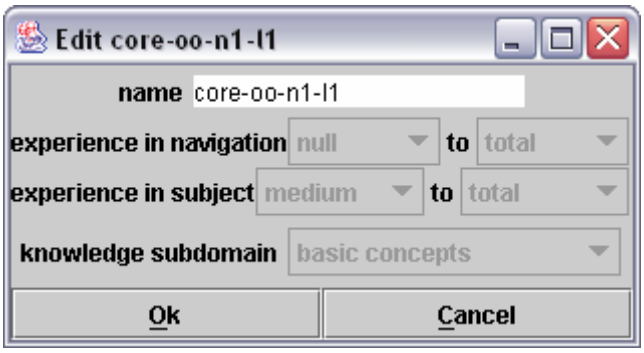
Paso I: Seleccionar el modo de navegación deseado pulsando sobre el icono correspondiente. Al hacerlo aparece una nueva ventana (*SEM-HP browser*), que permite navegar la estructura en el modo elegido.

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 (definidas anteriormente).



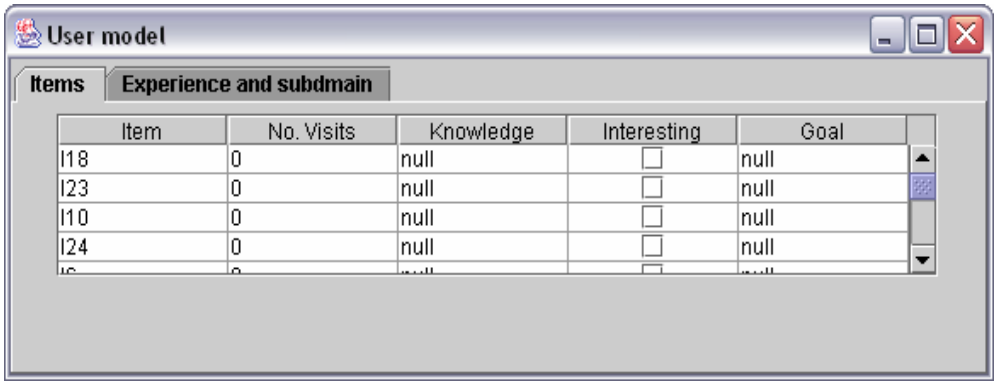
Paso II: Si el usuario desea ver las propiedades de la estructura elegida puede pulsar el botón *view CS attributes* del menú superior.



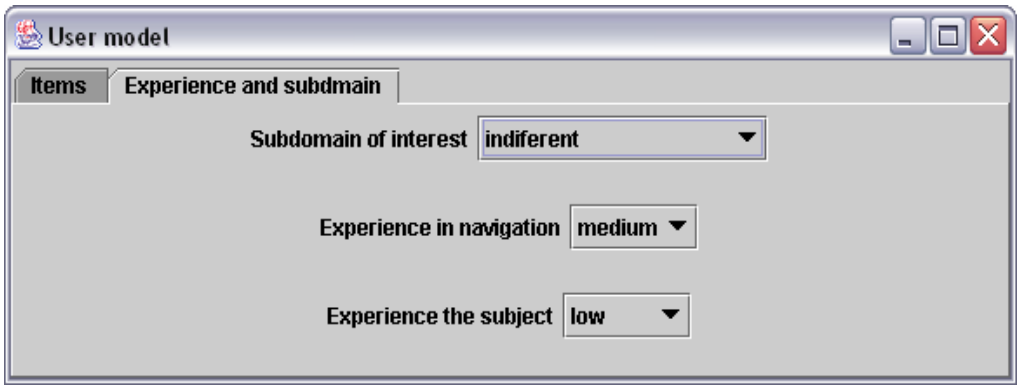
Observe que el etiquetado de la estructura se muestra, pero no se permite modificar (la funcionalidad de las listas desplegables se ha desactivado). Para hacer cambios es necesario trabajar en modo autor .

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.



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.

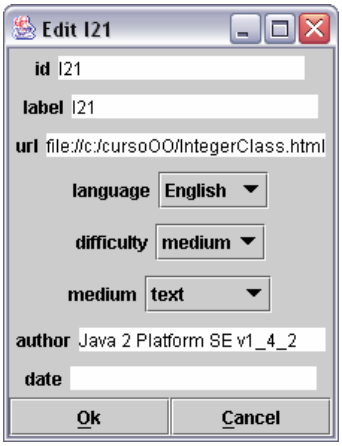


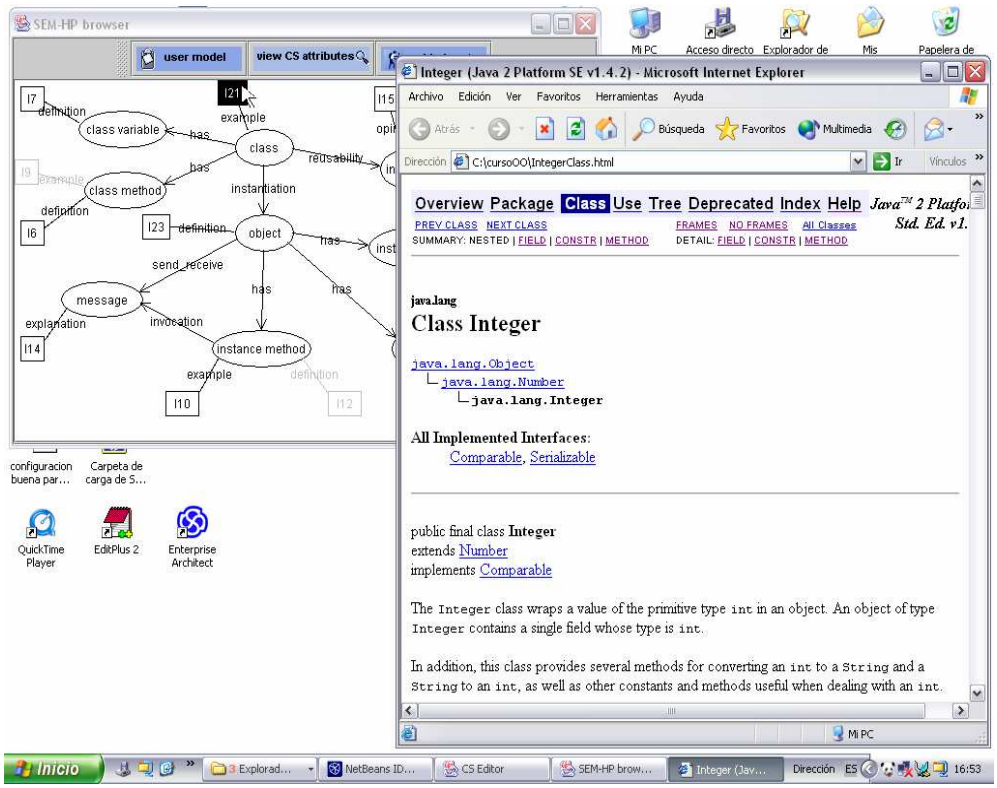
Paso IV: Para navegar, el usuario sólo tiene que hacer un *clic* sobre el ítem que desea visitar. Si éste es accesible, inmediatamente la información asociada al ítem se muestra en una ventana del navegador. En *Windows* se abre el navegador establecido por defecto, y en *Linux* se abre *Netscape*.

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 que se muestra más adelante, se ha seleccionado el ítem I21, cuyas propiedades se editan en la figura siguiente. Observe que tras ejecutar la regla de actualización por defecto Ru(I21), el conocimiento del usuario sobre I21 es "total" (aparece en negro).



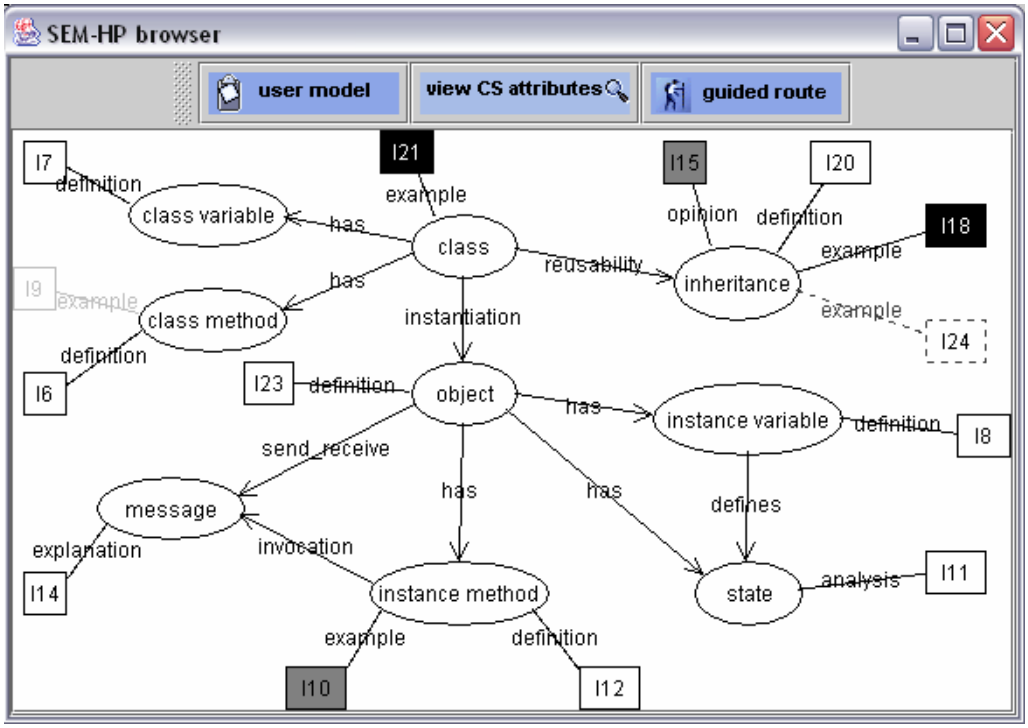


En este caso el documento mostrado se encuentra en el PC desde donde se ejecuta JSEM-HP (protocolo de conexión *file*). Por supuesto, también puede tratarse de un documento disponible a través de Internet (protocolo de conexión *http*).

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 muestra el estado del *browser* después de que el usuario ha visitado una vez el ítem I10. Su regla de actualización (definida anteriormente) 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".

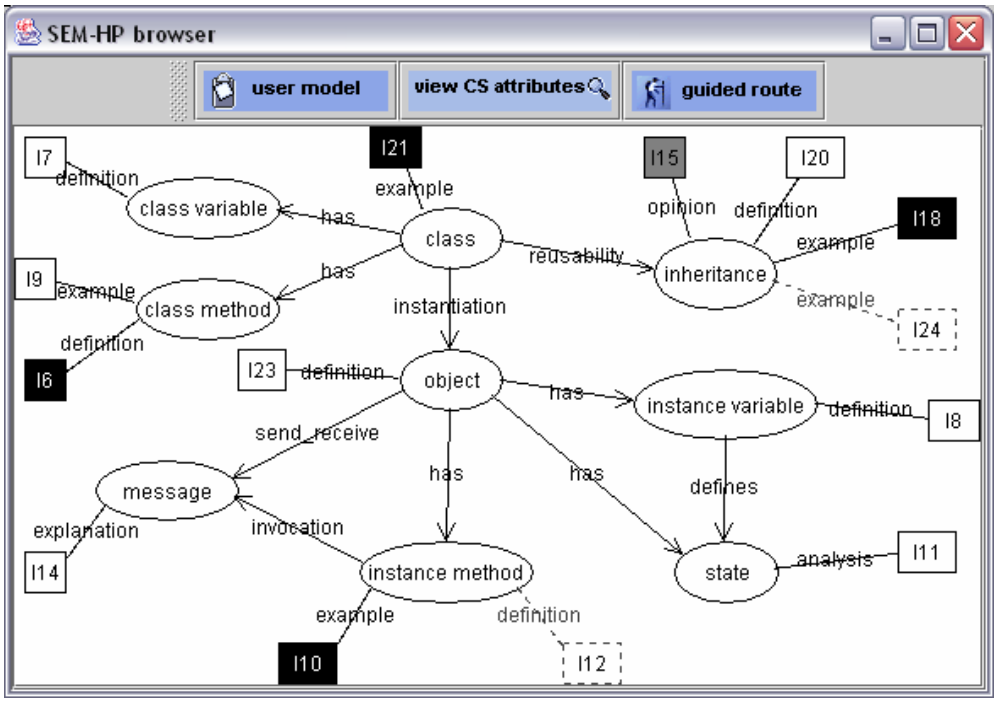
Con el nuevo estado, el ítem I12 se vuelve accesible por que el conocimiento sobre I10 supera el mínimo establecido en la regla Rk^1(I12) (definida anteriormente) . Y, el ítem accesible I24 deja de ser idóneo, porque el conocimiento sobre I18 ya no es inferior al conocimiento máximo establecido en la regla Rk^1(I24) (también definida anteriormente). La no idoneidad de I24 se refleja con una línea discontinua dibujando la asociación funcional y el borde del ítem.



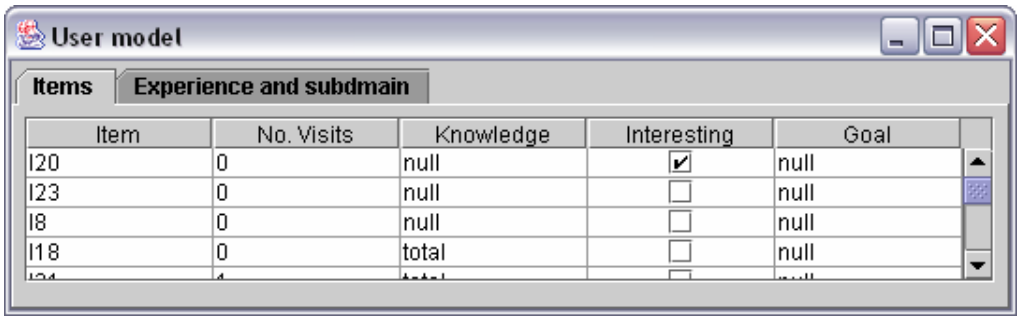
En la figura se muestra la anotación realizada, después de que el usuario seleccione a partir de la estructura de navegación mostrada en la figura anterior los ítems I10 e I6.

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) (definida anteriormente).

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) (también definida anteriormente), por lo que éste aparece como accesible e idóneo.



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. 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 siguiente).



PUBLICACIONES SELECCIONADAS

Applying software evolution theory to hypermedia systems

Fernando Molina-Ortiz

Department of Software Engineering
University of Granada
E.T.S.I.I., L.S.I.
Granada-18071, Spain
Fax: +34 958 243 179
E-mail: fmo@ugr.es

Lina García-Cabrera

Department of Informatics
University of Jaén
Campus Las Lagunillas
A-3, 132, Jaén-23071, Spain
Fax: +34 953 212 472
E-mail: lina@ujaen.es

Nuria Medina-Medina*

Department of Software Engineering
University of Granada
E.T.S.I.I., L.S.I.
Granada-18071, Spain
Fax: +34 958 243 179
E-mail: nmedina@ugr.es
*Corresponding author

Abstract: Environments on the web should support high-level hypermedia features to improve their key challenges: searching, navigation, adaptation and maintenance of hyperdocuments. In this paper, we show the benefits of applying software evolution theory to Hypermedia Systems (HSs) and, in order to face these challenges we propose the Semantic and Evolutionary Model for HyPermedia systems (SEM-HP) model. This model assumes that the development and maintenance processes of HSs are evolutionary; that is, they require a process of continuous change. These evolutionary mechanisms, in addition to other techniques (such as semantic representation, separation of concerns and user adaptation), allow the author to structure the hypermedia, select a presentation, design its navigation and specify adaptation rules according to his/her behaviour while he/she navigates. At the same time, an HS developed according to this model eases the search, supports and improves the navigation and adaptation and allows the evolution of the HS.

Keywords: hypermedia systems; user adaptation; software evolution.

Reference to this paper should be made as follows: Molina-Ortiz, F., García-Cabrera, L. and Medina-Medina, N. (2009) 'Applying software evolution theory to hypermedia systems', *Int. J. Web Engineering and Technology*, Vol. 5, No. 1, pp.69–87.

Biographical notes: Fernando Molina-Ortiz received his MSc in Computer Science from the University of Granada (Spain), initially worked at this university as a Researcher and now works as a Lecturer and Researcher. His research interests include the areas of software evolution and adaptive hypermedia systems.

Lina García-Cabrera is a Full Professor at the University of Jaén (Spain). She is also a member of the group on Software Development and Evolution at the Departamento de Lenguajes y Sistemas Informáticos (University of Granada, Spain). She received her PhD from this university. Her publications and research are in the areas of software systems evolution models and their application to the elaboration of authoring hypermedia tools for educational purposes.

Nuria Medina-Medina received her MSc in Computer Science in 2000 and is now a Professor at the Departamento de Lenguajes y Sistemas Informáticos of the University of Granada (Spain). She received her PhD from this university, with the thesis entitled 'An integral evolutionary model of adaptation for hypermedia systems: SEM-HP's learning system'. At the moment, she continues with her work in hypermedia systems, software evolution and user adaptation.

1 Introduction

The web has decisively contributed to the worldwide popularisation of hypermedia techniques. Nevertheless, the hypermedia community is pointing out the web's deficiencies and theoretical and practical inconsistencies. In fact, web systems can be seen as a subset of Hypermedia Systems (HSs), which work in the specific web environment (Díaz *et al*., 2005). In this sense, authors such as Bieber *et al*. (1997) demanded that web developers include the high-level features (typed nodes and links, link attributes, structure-based query, transclusions, warm and hot links, *etc.*) which are part of what they call fourth-generation hypermedia. Berners-Lee *et al.* (2002) proposed an extension of the current web, the semantic web, where "computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning".

Although several authors have proposed reference models with the aim of guaranteeing that HSs are correctly designed (for example, the Trellis metamodel by Stotts and Furuta (1989) or the Dexter model by Halasz and Schwartz (1990)), the web, from an abstract point of view, is based on a basic model (Rivlin *et al*., 1994) divided into a data submodel and process submodel. This model is general and flexible but incomplete. The data submodel does not define any constraints for determining the consistency of the information network and the process submodel is primitive since it does not detail the mechanisms to define the navigation.

We believe that it would be useful to build an environment that allows the development of hypermedia applications based on a given hypermedia model that characterises the information and determines how it can be organised, presented, navigated and adapted. Therefore, the model should have mechanisms to characterise the information (semantically structuring it), restructure the information and present, navigate and adapt it, with all of this done in a consistent way (García-Cabrera *et al.*, 2002). Finally, the web must adapt the hypermedia's presentation and navigation to the particular features of their users (De Bra *et al.*, 2004). In this research field, different models and architectures for Adaptive Hypermedia Systems (AHSs) have arisen. As a reference, we can cite the Adaptive Hypermedia Application Model (AHAM) (De Bra *et al.*, 1998) and the Adaptive Hypermedia Architecture (AHA) based on it (De Bra *et al.*, 2003). All these contributions try to provide HSs with more sophisticated environments that are able to face four important web challenges: the search for information, the design of navigation and adaptation and the maintenance of the information network.

With the same objective, we present the Semantic and Evolutionary Model for HyPermedia systems (SEM-HP) model, a semantic, systemic and evolutionary model for the development of AHSs. This model aims to provide support for evolution and user adaptation in a semantic HS. The structure of the paper is as follows: Section 2 briefly describes the relevant contributions to each challenge. In Section 3, we briefly describe the objective of software evolution theory. Section 4 introduces the SEM-HP model and Sections 5, 6 and 7 further describe it regarding the creation, execution and evolution of AHSs, highlighting its contributions to what we call the 'web challenges'. Finally, the related work, conclusions and further work are presented.

2 Web challenges

In this section, we show the most relevant contributions to different web challenges and their shortcomings, proposing at the same time several mechanisms that aim to reduce those shortcomings. Specifically, in the following subsections, we discuss searching, navigation, adaptation and maintenance.

2.1 Searching

It is evident that the web takes active part in this area; we can use powerful search engines. Although search engines can index an increasingly vast number of webpages, sometimes the search results for a given query are not appropriate, so there are pages that could be interesting for a user but are not accessed because they are not returned as the result of the query. More recently, search engines utilise Extensible Markup Language (XML). This language allows search engines to efficiently index data about websites without requiring a complicated crawler. The websites can simply provide an XML feed, which the search engines index. XML provides a universal medium for data or information exchange and for updating pages dynamically. Nevertheless, "XML allows users to add arbitrary structure to their document but says nothing about what the structures means" (Berners-Lee *et al.*, 2002); that is, it does not impose semantic constraints. The semantic web's challenge is to provide a language that expresses both

the data and the rules to reason about this data and allows to export to the web the rules of any existing knowledge representation system. It is necessary to formally define the relations among terms. By means of ontologies, the semantic structure is made explicit so we can improve the accuracy of web searches and relate the information on a page to the associated knowledge structures and inference rules.

2.2 Navigation

Navigation is the origin of the power, but also of the problems of HSs. By browsing, we can explore an information system in a nonlinear and flexible order. But, at same time, this flexibility produces disorientation and a cognitive overhead. In a hyperdocument, the distances are not normal: all the links are equally long. There is no way to know if a link is near to a previously visited document and, consequently, we end up lost in hyperspace. Navigation in the web, although eased by some support tools (highlighted links, history list, bookmarks, *etc*.), is guided by the user and not by the system. In addition, these aids only assist the users in searching for information, but not in navigating with the aim of understanding the information.

Maybe we have forgotten what Bush wrote in his famous paper 'As we may think' (Bush, 1945):

> "Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. ...The human mind does not work that way. It operates by association."
>
> "The process of tying two items together is the important thing."

With the same idea, Kibby *et al.* (1990) and Mayes (1994) identified two parallel structures in a hypermedia network. One of them provides the set of explicit nodes and links (the hyperspace). The other structure references the conceptual space that is sketched from the explicit and implicit knowledge present in the nodes, links and in all the possible associations among the information. Browsers help the user explore the hyperspace, but they ignore the conceptual space that is developed in parallel. The conceptual space is much more complete than the hyperspace because it is also based on implicit knowledge. Hyperspace has only to search for and find information, while conceptual space has to achieve an understanding of that information. That vision is fundamental in systems related to teaching-learning and in any website that offers education.

Therefore, to design an HS in which the user accesses information resources just by the association of ideas, it is necessary to represent the conceptual space. In addition, the final structure of a hyperdocument could be determined by:

- the Conceptual Structure (CS) of the provided information
- the access modes and the way in which this information (or part of it) is going to be browsed.

As well as representing the CS, we need to separate the concerns on the development of HSs:

- creation of conceptual and information systems
- selection of presentations or views of them

- design of the navigation through these presentations
- taking care of user adaptation.

2.3 Adaptation

AHSs deal with user adaptation. In these systems, navigation becomes a shared responsibility between the user and the system. In this case, an inference mechanism is used to create paths according to the user's behaviour. There are well-known contributions in this field, such as those of De Bra *et al.* (1998; 2003), Brusilovsky (1996) or Koch and Wirsing (2002). In all of them, user adaptation modifies user navigation according to the user's specific features, but it does not perform structural changes in the system. Functional changes are carried out without requiring the direct intervention of the developer, using adaptive methods defined previously by him/her. But to perform structural changes and establish rules that allow specifying the navigation, an explicit representation of the conceptual space is necessary.

Many contributions aim to provide a clearer distinction between conceptual representation, presentation, navigation and adaptation. Again, the separation of concerns provides an incremental development of adaptation. Besides, adaptation models focus on modifying the form and functionality of the provided navigation structure, but few of them offer different ways to navigate it. They offer alternative navigation structures which are navigated with the same philosophy. A navigation and adaptation scheme that takes into account the conceptual space and user knowledge will improve the comprehension of the information system, reduce disorientation and the cognitive overhead and could allow different ways of navigation that the user can choose according to his current needs. Finally, in the existing models, there is no explicit support for changing the predefined adaptive methods. For this, evolutionary mechanisms are needed that guarantee the consistency of changes in the rules of adaptation.

2.4 Maintenance by evolution

Maintenance can be considered one of the most overlooked issues in HSs. Common sense says that the web's content without any knowledge representation is designed for humans to read, not for computer programs to manage and support.

Again, we need the information to be based on a conceptual space upon which we can establish some logic rules that the system applies to support the development of the HS. To sum up, an HS must be a special kind of information system based on a conceptual space. This knowledge representation cannot be universal and can change. We need tools to construct this semantic representation and rules that say how we can structure this knowledge. In our opinion, the development process of an HS, as it is with any Software System (SS), is iterative and evolutionary. A good separation of concerns both in the development (conceptual representation, presentation, navigation and adaptation) and evolution processes facilitate the further development and maintenance of the information system.

3 Software evolution theory

As we have previously discussed, we can make progress in web challenges by means of: knowledge representation, an explicit semantic representation of the conceptual domain of the HS and the separation of concerns on the development process of an HS.

For us, the development process of an HS is iterative and evolving in nature. It is necessary to provide mechanisms of change that permit modifying and improving the HS. In software evolution theory (Parets and Torres, 1996), an SS is a set of systems which interact between themselves and with the environment. These systems have a structure that offers functionality. The developer (or author) performs changes in the SS during its construction (modifying its structure), but also later during its functional life. These later changes modify the structure and/or functionality of the system to produce adaptations which guarantee the usefulness of the interaction of the SS with the environment.

In the evolutionary process of an SS, the developer is a very important element, since he/she is in charge of designing the capacity of evolution of the SS and, after that, carries out evolutionary actions (ACe) to produce the necessary changes over the system. Another important element is the metasystem (Parets and Torres, 1996), which conducts the interactions between the SS and the developer. The SS suffers the changes and finally, the user utilises it.

From a general perspective, the SS can evolve in two different ways:

1 evolution driven by the developer

2 self-evolution of the SS.

The first type of evolution (structural changes) implies a direct intervention of the developer in that he/she is the one who drives the changes in the system. The second type (functional changes) is performed in an automatic way depending on certain mechanisms defined previously by the developer and the behaviour of the current user; therefore, the developer's intervention in this second case is indirect. We can consider user adaptation a special case of this second type of evolution (Medina *et al.*, 2002).

4 The SEM-HP model

Based on previous works in software evolution, we have presented the SEM-HP model. It supports both the development and evolution of AHSs, since changes are only allowed when they lead to a consistent system. Besides, SEM-HP has been improved to take user adaptation into account. SEM-HP's architecture (Figure 1) makes a double separation of concerns: vertical (into four subsystems) and horizontal (system and metasystem). The vertical division structures the AHS in four interrelated subsystems, each of which offers a different functionality and is represented by a model that supports that functionality. Table 1 summarises the main aspects of the subsystems in SEM-HP, which will be detailed in this paper:

- Memorisation Subsystem (MS) – semantically defining, structuring and maintaining the concepts and information offered by the system
- Presentation Subsystem (PS) – selection of subsets of the elements offered by the MS, which we will call presentations

- Navigation Subsystem (NS) – defining the paths that the user can follow while he/she browses a presentation
- Learning Subsystem (LS) – adaptation of the navigation to the specific features of each user (user adaptation) and analysis of the behaviour of the user group to suggest adjustments in the other subsystems (feedback). One of its tasks is modelling how users learn.

Figure 1 The SEM-HP architecture (see online version for colours)

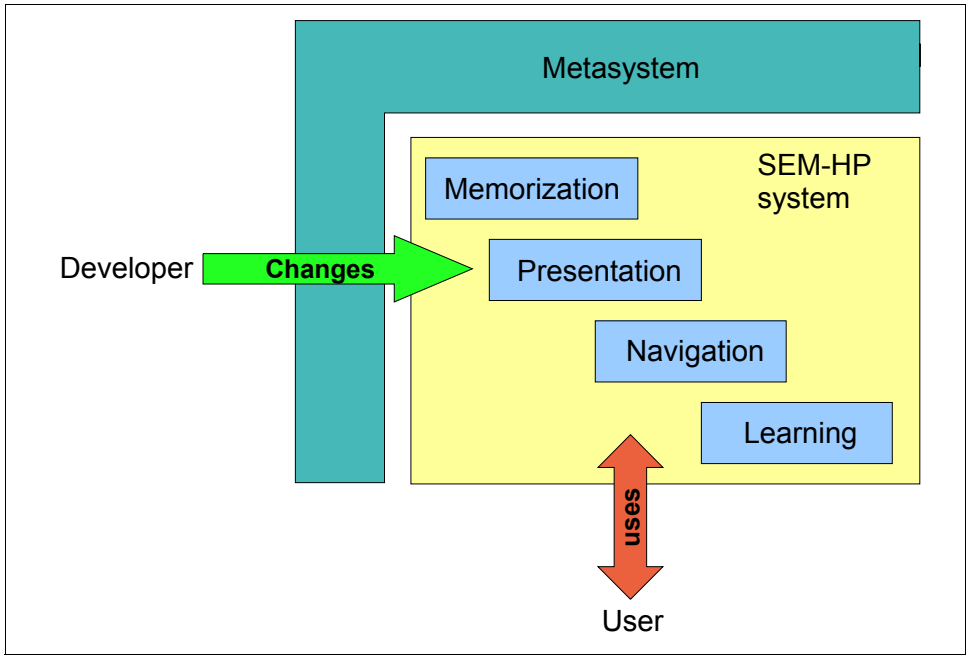


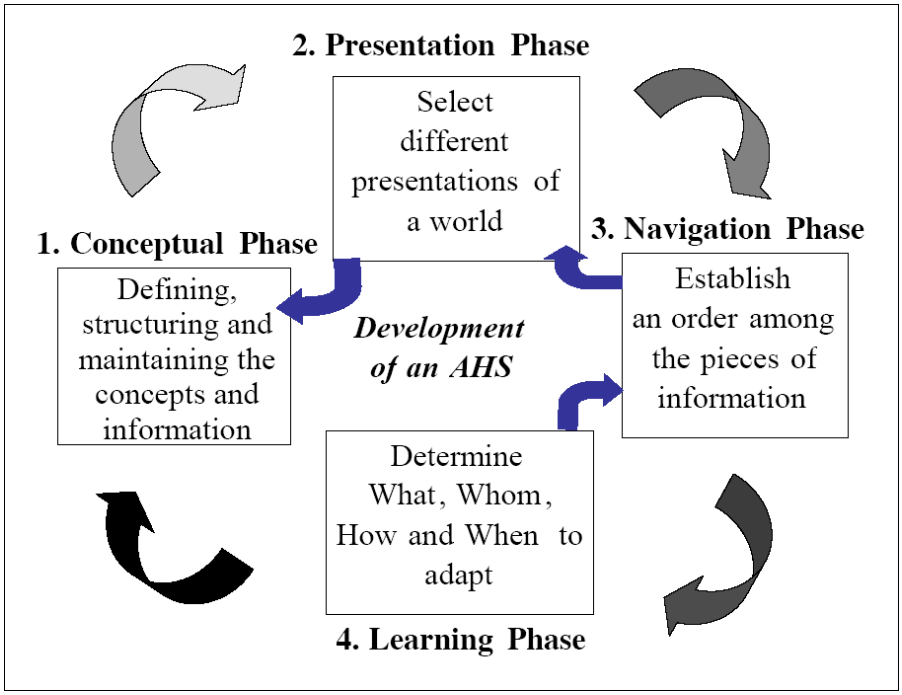
Table 1 Features of the subsystems in SEM-HP

Feature	*MS*	*PS*	*NS*	*LS*
Model	Conceptual structure	Conceptual structure	Order rules	User model and weight, update and knowledge rules
Representation	Semantic net	Semantic net	Propositional temporal logic	Tuple space and predicate logic
Execution	–	–	–	Four types of navigation
Evolution	✔	✔	✔	✔
Propagation to	PS	NS	LS	–
Feedback to	–	–	–	MS, PS

The horizontal division distinguishes two abstraction levels: system and metasystem. The metasystem assists the author in the task of evolving each representation model according to the requirements of the environment and guarantees the co-evolution of the subsystems as a whole (automatic propagation of changes from one system to the other systems).

SEM-HP also proposes a development process with four iterative phases, each of which corresponds with one of the four subsystems introduced below. These phases are shown in Figure 2 and, excluding user adaptation, are conceptually similar to those proposed in many other methodologies, such as Object Oriented Hypermedia Design Model (OOHDM) (Schwabe *et al.*, 1996) or Web Modelling Language (WEBML) (Ceri *et al.*, 2002).

Figure 2 The development phases in SEM-HP (see online version for colours)

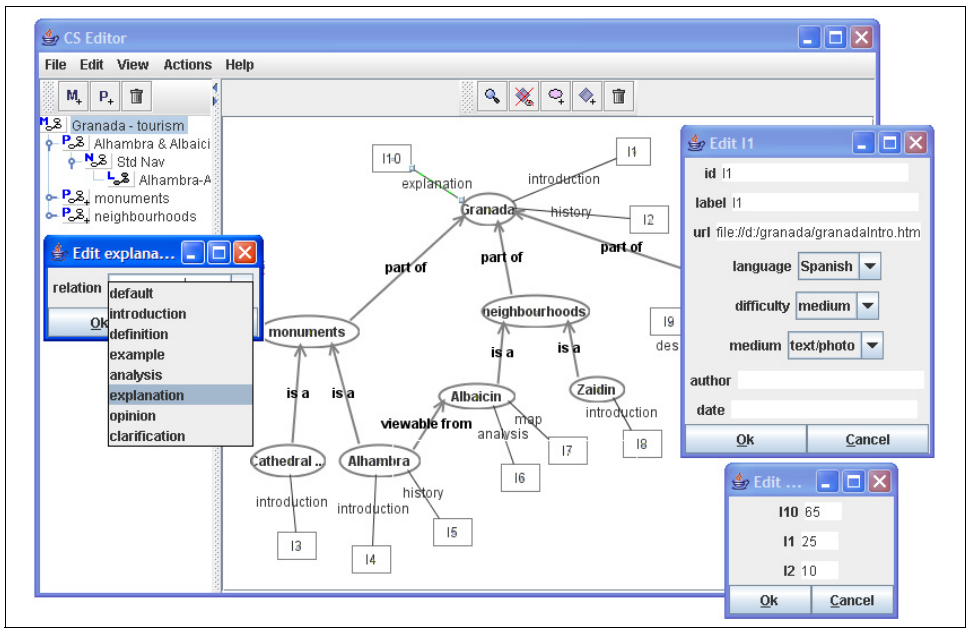


In addition, we propose an author tool called JSEM-HP (the name comes from Java, the language in which it is implemented). It presently implements the main features of SEM-HP and was described by Molina *et al.* (2006). Figures 3, 4 and 6 have been done with this author tool.

4.1 Memorisation subsystem

The MS is in charge of storing, structuring and maintaining the information offered by the HS. For that, the semantic structure is made explicit by the model by means of a CS. The CS is formalised through a semantic net, in which there are two kinds of nodes: concepts and items. The concepts are ideas, thoughts or abstractions and are interrelated through conceptual associations. The items are pieces of information offered by the HS, which are joined to concepts by functional associations. The functional association represents the role that the item has in describing the concept (explanation, example, description, *etc.*). An example of a CS about the city of Granada, as created in JSEM-HP, can be seen in Figure 3. Concepts are represented with ovals, items with rectangles, conceptual associations with arrows and functional associations with lines.

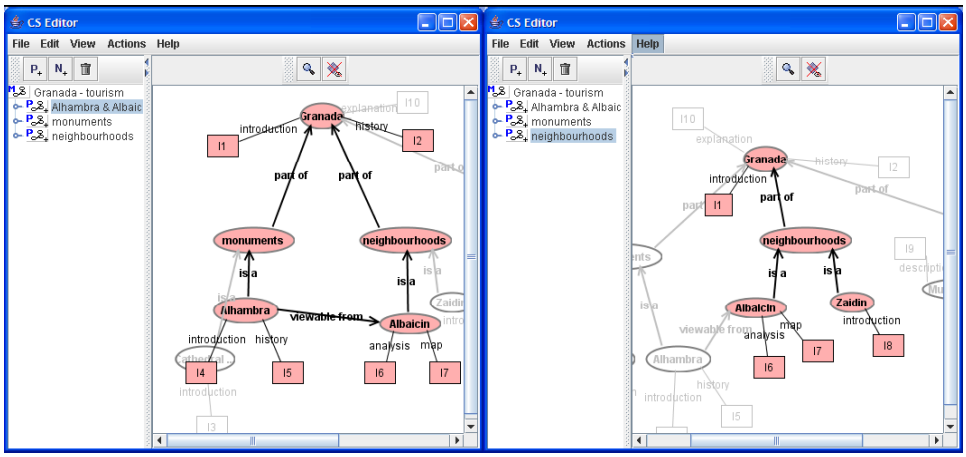
Figure 3 The JSEM-HP creating a CS about the city of Granada (see online version for colours)



4.2 *Presentation subsystem*

The PS allows the author to prepare different views of the same information, creating personalised CSs (CS_P or Conceptual Structure of Presentation). To do this, he/she filters the CS provided by the MS and reduces the concepts, items and associations present in it. The author can also change the spatial arrangement of the concepts and items in the presentation. Figure 4 shows two CS_P created from the CS of Figure 3.

Figure 4 Two possible presentations (see online version for colours)



4.3 Navigation subsystem

The NS establishes the order in which the user will be able to visit the information pieces in each CS_P, creating the corresponding Conceptual Structure of Navigation (CS_N). The possible paths are represented using order rules (Ro), which are formalised using temporal propositional logic. The body of an Ro establishes which items the user must visit to be allowed to access the item in the head of the rule and specifies which conceptual relation is followed. In the body of the rule, the items can be joined using the logical operator opL = {and, or} and visiting each of them is restricted with a temporal operator opT = {previous, before, not before}. The previous operator, when applied to an item, requires that the item must be visited just before the head item and the 'before' operator requires the item to be visited at least two steps beforehand. Again, it is possible to define different navigations for the same presentation.

4.4 Learning subsystem

The LS (Medina *et al.*, 2005) is in charge of adjusting the navigation process to the features and interests of the current user. To do this, it builds and updates a user model represented as a set of tuples. Each tuple stores one user attribute regarding his personal characteristics, experience, knowledge, preferences and goals.

For example, for each item and concept in the CS, the user model stores the degree of knowledge (K) that the user has about it. Therefore, it is able to restrict the navigation of the user depending on his/her knowledge and not only on the items that he/she has visited. The K of the user is represented by a semantic label chosen among five possible labels ('null', 'low', 'medium', 'high' and 'total').

The user model is updated while the user navigates through the weight rules (Rw) and update rules (Ru). The knowledge rules (Rk) determine which items can be visited depending on the user's knowledge. Again, thanks to the vertical division in subsystems, the author can establish different sets of these rules for the same CS_N, defining diverse adaptation criteria in different Conceptual Structures of Learning (CS_L), which include a set of weights, Rk and Ru.

The Rw are simple mathematical formulae that calculate the user's knowledge about a concept c_a from the user's knowledge about the items that are associated to the concept in the CS. The term associated to an item i_j is weighted with the influence p_j that the knowledge about that item, $K(i_j)$, has on the knowledge about the concept, $K(c_a)$.

The Ru are if-then rules that model how the user learns by updating the user's knowledge about the items of a concrete CS_L. When the user visits one item, his/her $K(i_j)$ and possibly about other items is updated in the user model by means of the Ru defined by the author. The if-side of the rule is a logic predicate $Visit(i_j)$ that becomes true each time the user visits item i_j and the then-side is a set of predicates in which each predicate updates the user's knowledge about a different item. An update can be incremental or fixed (the knowledge about an item can increase or be set to a fixed degree), absolute or relative (the predicate can use an absolute semantic label or be based on the degree of user knowledge about the visited item), first-time or each-time (the update can be run only on the first time the item is visited or each time it is visited).

Rk are represented using predicate logic. The Rk associated to an item i_j in a CS_L determines what other items are its pedagogic prerequisites, establishing the needed K about them to reach item i_j in optimal conditions of learning. Hence, the user must satisfy several knowledge restrictions, each of which affects a different item and requires a knowledge level that is greater or lesser than what was specified. In the same CS_L, the author can define several Rk for an item if there are different ways to get ready to learn it.

5 Creation of an AHS in SEM-HP

To ease the task of the author during the development of the AHS, SEM-HP only requires him to build at least the CS in the MS, since the system can generate from it the representation models for other subsystems. The initial steps performed to create an AHS with SEM-HP are as follows:

Step 1 *The author specifies a CS*: The author creates concepts and links them using conceptual relations. There are predefined relations (partOf, isA or aKindOf), but if the author needs it, the system allows new conceptual relations to be defined. Besides, the author associates items of information to the created concepts, stating in each association the function or role the item is playing (see the possible roles in the CS shown in Figure 3).

Step 2 *The system creates a presentation*: By default, the generated CS_P is equal to the initial CS. It incorporates all the concepts, items and associations that are included in the CS created by the author.

Step 3 *The system generates the Ro*: For each item in the CS_P, the system generates an Ro for every conceptual relation that arrives to the concept to which the item is associated. In this way, in the body of the rule, the system specifies that any of the items associated to the origin of the conceptual relation must be visited immediately before the head item. The default Ro associated to item I6 in Figure 3 is:

follow("viewable from"): previous(I4) or previous(I5) → I6.

Step 4 *The system generates the weight, Ru and Rk*:

Step 4.1 For each concept in the CS, the system generates an Rw in which all the items associated to the concept have the same weight. For example, the default Rw for the concept Alhambra in Figure 3 is:

K(Alhambra) = 1/2 K(I4) + 1/2 K(I5).

Step 4.2 For each item in the CS_P, the system generates an Ru that performs only an update when the item is visited. This update affects that item and is fixed and absolute (predicate Fix-abs). Specifically, it establishes that the degree of user knowledge about an item after visiting it is 'total'. The default Ru for the item I5 is:

Ru(I5): Visit (I5) → Fix-abs (I5, "total").

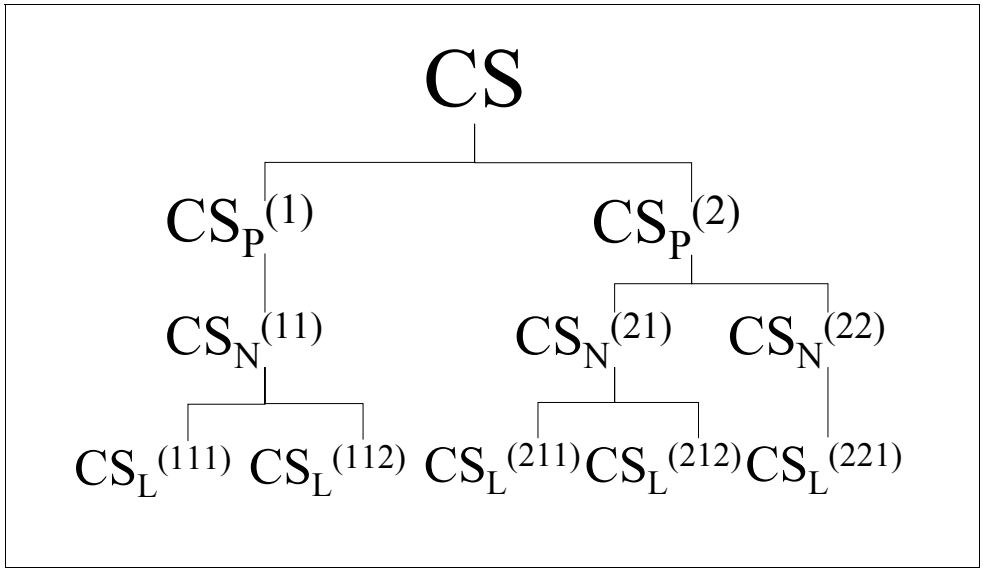
Step 4.3 The system obtains an Rk from every Ro. To do this, it places a knowledge restriction in the Rk for each previous requirement in the body of the Ro. This restriction, which is applied to the same item affected by the previous requirement, requires knowledge larger than 'null':

Rk(I6): K(I4) ≥ "null" or K(I5) ≥ "null" → I6.

6 Execution of AHS in SEM-HP: browsing by the user

As we have seen, each subsystem is incrementally defined based on the previous one so the author can define from a CS different CS_P, from each of these CS_P he/she can define different CS_N and from each CS_N he/she can define different CS_P (for example, Figure 5 shows five different CS_L). As described in Medina *et al.* (2005), the system can automatically select the CS_L that best fits the user profile.

Figure 5 The incremental definition of subsystems



In SEM-HP, the user browses directly a CS and when he/she clicks on an item, it opens a window that shows the item's content. In addition, SEM-HP offers to the users four types of navigation depending on their objective: free navigation by items, free navigation by concepts, navigation restricted by conceptual relations and navigation restricted by knowledge (Medina *et al.*, 2005). When the objective is to learn pieces of information, the last type is chosen, which restricts user navigation as specified by the Rk. The CS is obtained from the CS_P associated to the chosen CS_L and it is adapted to the user by disabling the items for which the Rk are unsatisfied by the user's current knowledge, thereby presenting to the user only the items he/she is ready to understand. Because of this adaptation process, not all the items in the CS are always available, since those whose knowledge restrictions are not satisfied by the user are hidden and disabled. The items

that have been previously visited by the user and the interesting items for the user's goal are shown in different colours. Besides, both items and concepts are annotated according to the degree of user knowledge (darker as his knowledge increases), so he/she can see how his/her knowledge grows as he/she accesses the information offered by the system. Figure 6 shows an example of browsing after visiting item I1. In addition to the CS, there is a web browser window with the contents of the item.

Figure 6 JSEM-HP: navigation by the user (see online version for colours)

While the user navigates, the LS updates the user model (by applying Ru) and uses the mentioned adaptation techniques (links annotation and hiding) in the CS_L. Besides these, the LS supports other forms of adaptation, such as guided routes (Medina *et al.*, 2003), personalised views, orientation support and feedback. Adaptation by feedback aims at the convergence of the representation models created by the author and the mental models that the users of the system have (Medina *et al.*, 2006).

7 Evolution of AHS in SEM-HP

In the previous section, we described how the system builds a complete AHS starting with the conceptual and informational domain provided by the author. However, thanks to its two-level architecture, SEM-HP supports the evolution of the created system, allowing the author to change any of the four subsystems in the AHS. To perform these changes, the system provides a set of ACe. The author tool will generate the appropriate ACe for the change that the author chooses to perform. An ACe is only accomplished if it satisfies a set of restrictions that assure the consistency of the change. There are two kinds of restrictions: the restrictions imposed by the system (called 'invariants' because they are fixed) and the author restrictions. The author can change the author restrictions using a special set of ACe whose restrictions are called 'metarestrictions' (restrictions on restrictions).

Now we summarise the changes that the author can perform in the representation model of each subsystem: in the MS, the author can create, modify or remove concepts, items and functional and conceptual associations of the CS. In each CS_P of the PS, the author can hide or show the concepts, items and conceptual and functional associations of the initial CS, effectively selecting a subset of the CS to be presented. In addition, he/she can define new CS_P from the CS. In the NS, for a CS_N, the author can enable or disable Ro, change the opLs of an Ro or extend an Ro by adding new restrictions to the body of the rule. The restrictions added to an Ro by the author can be also removed by him/her. Some modifications that can be done to the Ro in Step 3 in Section 5 are shown in boldface below:

follow("viewable from"): **before (I3) and** [previous(I4) or previous(I5)] → I6.

In the LS, the author can create or modify a CS_L: he/she can change the weights in Rw, add, delete or change the update predicates in Ru or change the knowledge restrictions in the body of Rk. Some modifications that can be done to Rw and Ru in Step 4 in Section 5 are shown in boldface below:

K(Alhambra) = **1/3** K(I4) + **2/3** K(I5)

Ru(I5): Visit (I5) → Fix-abs (I5, "total"), **Inc-abs (I4, 1 level up**).

Each of the cited ACe has a set of restrictions which guarantee consistency after the change. For example, disconnected concepts or different concepts with the same name are not allowed in the CS, all the Ro associated to an item cannot be disabled, the update to the visited item cannot be removed in an Ru, or a set of rules Ru ∪ Rk that would cause any item to be always unreachable for the user is not allowed.

Sometimes, when an element in a model is changed, the need to perform new changes in other elements of the same subsystem (internal propagation) or even in elements of another subsystems (external propagation) arises. Both kinds of change propagation are supported in SEM-HP through the metasystem, which runs the necessary changes to perform propagation without the author's intervention. Internal propagation solves the consistency problems inside each subsystem; for example, since disconnected concepts are not allowed in the same CS, when removing a concept, all the associations starting or ending with it are also removed, as are all the concepts which get disconnected as a consequence of the change.

The external propagation mechanism allows all the subsystems to co-evolve in a consistent way after the changes performed by the author. That is, when the author changes a representation model, the metasystem checks if the performed changes in this model imply new changes in any of the other models and, if they are necessary, runs the appropriate ACe to carry them out. Because of this, in addition to the individual consistency of each subsystem, the metasystem also guarantees the global consistency of the AHS.

For example, the modification of the Ro described in the previous section would cause propagation from the NS to the LS. The changes propagated to the Rk in the LS are shown in boldface below:

Rk(I6): [**K(I3)>"null" and** K(I4)>"null"] or
[**K(I3)>"null" and** K(I5)>"null"] → I6.

Some changes do not require external propagation (such as the ACe carried out in the LS), while some ACe can give rise to many changes during the external propagation process. For example, when removing an item in the CS of the MS, it will be necessary to remove that item in all the presentations that have it. This, in turn, causes the removal of the item from the body of all the Ro, Rw, Ru and Rk in which it appears. Besides, all the Ro, Ru and Rk where the removed item is the head item must be removed. Hence, this change affects the four subsystems.

8 Related work

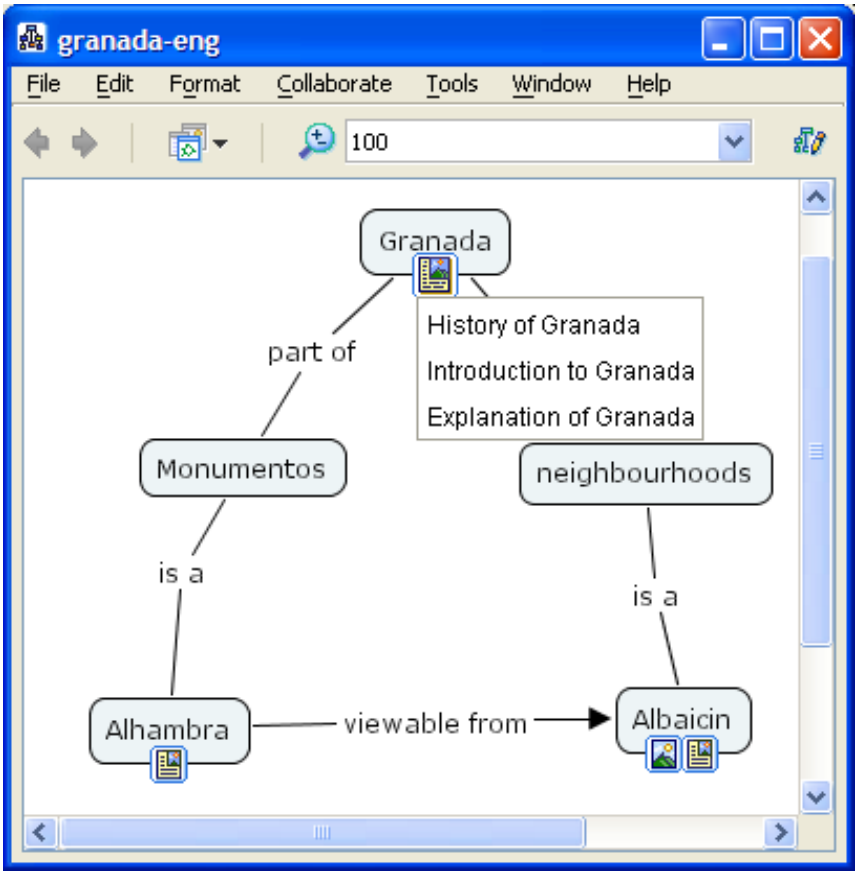
Several architectures and models for the development of HSs have been proposed. A good review of the software engineering approaches to modelling HSs is presented in Lowe and Wall (1999). Among the different models we can find in the literature, we can cite the Reusable Intelligent Collaborative Hypermedia system (RICH) model (Wang and Rada, 1998), which is oriented towards writing and reusing scientific-technical documents. It uses semantic nets to represent hyperdocuments, but it does not clearly distinguish conceptual and information domains. In addition, it does not allow the navigation of the created documents taking into account the semantic structure. Finally, RICH does not contemplate a complete development process for HSs and it does not allow the creation of HSs with a generic domain, since it is oriented towards the reuse of scientific hyperdocuments.

HMBS (Ferreira *et al.*, 2001) is a Hypermedia Model Based on Statecharts and the context-aware Trellis (caT) model (Furuta and Na, 2001) builds directly the Petri net-based document structure without constructing a CS. Both do not perform a clear separation of concerns and they do not support adaptation to user knowledge.

Regarding AHSs, we can cite, among other relevant work, the AHA! architecture (De Bra *et al.*, 2003). This architecture is an extension of AHA, based on the AHAM model (De Bra *et al.*, 1998) which, as SEM-HP, distinguishes a domain model, a user model and an adaptation model. Although they are conceptually similar in several aspects, there are also significant differences. One important difference lies in the way the hypermedia is browsed: while in SEM-HP, the user browses a CS (so the semantic structure is explicit), AHA! adopts a more conventional approach in which the user directly browses pages, so the orientation support provided by an explicit CS is lost. In both systems, there are rules that update the user model and are triggered by when the user visits elements in the HS, but AHA! allows defining any kind of update in the user model (knowledge, interest, *etc.*) while in SEM-HP, Ru are used only to update the user knowledge and interests and goals are defined in a different way. This may lead us to think that SEM-HP is more restrictive, but this is done with an idea in mind: evolution. A formal definition of, for example, Ru and Rk allow checking the consistency of the HS while it evolves; in this case, the metasystem checks (among other things) that the Rk and Ru will not lead to unreachable items. The evolutionary approach, the consistency of the different representation models and the integration among the different formalisms provided through change propagation may be seen as considerable novelties provided by SEM-HP.

Regarding knowledge representation, we would like to mention that the CSs in SEM-HP are similar to concept maps, initially described by Novak and Gowin (1984). Since in SEM-HP, the user directly navigates an explicit CS, a navigable conceptual map would have many things in common not only with knowledge representation, but also with the way the system is used. CmapTools (Cañas *et al.*, 2004) is a set of tools that allows building, navigating, sharing and criticising knowledge models represented as conceptual maps. Concept maps, as SEM-HP CSs, can be seen as semantic networks. In the case of concept maps, the only nodes are concepts, which are connected with directed labelled associations to form propositions. CmapTools additionally permits enriching concepts with resources, which can be links to pages, documents, images, *etc.* These resources correspond to SEM-HP's items and a significant difference is the way in which they are connected to concepts. Figure 7 shows a concept map equivalent to the CS shown in Figure 4 (CS_P on the left). Resources are accessed through the icons overlapping with concepts. There is an icon for each kind of resource (webpage, image, *etc.*) and clicking on the icon will bring a list of resources which will be opened if selected. We think that using functional associations that describe the role of the item is more intuitive than CmapTools' approach: it is more important for the user to know if an item is an example or a definition rather than if it is a webpage or a .pdf file.

Figure 7 CmapTools (see online version for colours)



Continuing with the knowledge representation in CmapTools, we would like to mention that CmapTools allows nesting concept maps within others, a feature that has been deliberately excluded in JSEM-HP because of the additional disorientation problems it could cause to the user when navigating from one conceptual map to another. Complexity in CSs is reduced by selecting a CS_p that is suitable to the user's features and goals and afterwards, by adapting it according to conceptual relations or the user's knowledge, depending on the type of navigation chosen. In addition, in SEM-HP, there is a type of navigation (free by concepts) where the system automatically generates a summary of each concept, composing the items of information associated to it according to their roles and the preferences specified by the user. If compared with JSEM-HP, CmapTools is a more mature application which has some interesting features lacking in JSEM-HP, such as the support for sharing and criticising conceptual maps. Nevertheless, CmapTools lacks the support for evolution and user adaptation present in JSEM-HP.

9 Conclusions and further work

In this paper, we consider that searching, navigation, adaptation and maintenance are key elements in HSs and, therefore, in the web. The analysis of these areas has allowed us to identify two fundamental mechanisms: explicit semantics and the separation of concerns.

An explicit semantic would allow searches that are more precise and automated, offers conceptual navigators that ease the understanding of hyperdocuments, improves the modes of adaptation and navigation and, finally, is fundamental for maintenance.

The separation of concerns eases the development and maintenance of the system. The proposed approach seeks to be independent of the technological platform; the separation of subsystems and levels (system and metasystem) allows the clarification of the development process and the automatic verification of changes and their propagation. Table 2 summarises how SEM-HP approaches these challenges.

Table 2 The web challenges from SEM-HP's perspective

	SEM-HP	
Web challenge	*Explicit semantics*	*Separation of concerns*
Searching	The conceptual structure permits more intelligent searches	
Navigation	The user navigates directly on a conceptual structure represented as a semantic net	The offered information is not mixed with how it is navigated (navigation rules are defined on top of the conceptual structure)
Adaptation	Better definition of adaptation rules	Different types of navigation and methods of adaptation
Evolution	Formal definition of evolutionary actions	Metasystem takes care of global consistency

SEM-HP is a model for the development of semantic AHSs that takes their evolution into account. Its associated development process and evolutionary approach allow the evolution of the HS while maintaining the desired consistency. The semantic focus ensures that the information offered is given a semantic coherence, which is translated

to the user via a navigable CS. Finally, the system can adapt to the user while he/she navigates, reduces the problems of disorientation and cognitive overhead and makes the user aware of his/her progress.

Currently, our work is focused on finishing and refining the implementation of JSEM-HP in order to validate its usefulness both in the creation and evolution of HSs and the reduction of the problems associated to nonlinear navigation by means of user adaptation. The usefulness of the SEM-HP model has already been verified in conceptual ways, such as in the integration of children with autism problems (Gea *et al.*, 2004). With the aim of validating it in a practical way, preliminary tests with the modelling part of JSEM-HP have been done. Although the tool allows building Rk and Ru interactively using only the mouse, we have seen that the authors still need too much knowledge about the underlying model. Therefore, the Rk have been simplified so a more intuitive user interface can be built. Once the tool is completely finished, a more thorough validation will be done.

Acknowledgement

This research is supported by the Spanish MCYT under R+D projects TIN2008-06596 -C02-02, TIN2007-64718 and TIN2007-60199, by the Research Programme of the Andalusian Government under project P08-TIC-03717 and by the European Regional Development Fund (ERDF).

References

Berners-Lee, T., Hendler, J. and Lassila, O. (2002) 'The semantic web', *Scientific American Special Online Issue*, pp.24–30.

Bieber, M., Vitali, F., Ashman, H., Balasubramanian, V. and Oinas-Kukkonen, H. (1997) 'Fourth generation hypermedia: some missing links for the World Wide Web', *International Journal of Human-Computer Studies*, Vol. 47, No. 1, pp.31–65.

Brusilovsky, P. (1996) 'Methods and techniques of adaptive hypermedia', *User Modeling and User-Adapted Interaction*, Vol. 6, pp.87–129.

Bush, V. (1945) 'As we may think', *The Atlantic Month*, Vol. 176, pp.101–108.

Cañas, A.J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., Gómez, G., Arroyo, M. and Carvajal, R. (2004) 'CmapTools: a knowledge modeling and sharing environment. Concept Maps: theory, methodology, technology', *Proceedings of the First International Conference on Concept Mapping*, pp.125–133.

Ceri, S., Matera, M., Bongio, A., Fraternali, P. and Comai, S. (2002) *Designing Data-Intensive Web Applications*, Morgan Kaufmann Publishers.

De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D. and Stash, N. (2003) 'AHA! The adaptive hypermedia architecture', *Proceedings of the ACM Hypertext Conference*, Nottingham, UK.

De Bra, P., Aroyo, L. and Chepegin, I. (2004) 'The next big thing: adaptive web-based systems', *J. Digit. Inf.*, Vol. 5, No. 1.

De Bra, P., Houben, G.J. and Wu, H. (1998) 'AHAM: a reference model to support adaptive hypermedia authoring', *InfWet'98 Conference*, http://wwwis.win.tue.nl/~debra/infwet98/paper.pdf.

Díaz, P., Montero, S. and Aedo, I. (2005) *Ingeniería de la web y patrones de diseño*, *Prentice Hall.*

Ferreira, M.C., Santos, M.A. and Masiero, P.C. (2001) 'A statechart-based model for hypermedia applications', *ACM Transactions on Information Systems (TOIS)*, Vol. 19, No. 1, pp.28–52.

Furuta, R. and Na, J-C. (2001) 'Applying caT's programmable browsing semantics to specify world-wide web documents that reflect place, time, reader, and community', *Proceedings of the 2002 ACM Symposium on Document Engineering*, McLean, Virginia, 8–9 November, ACM Press, pp.10–17.

García-Cabrera, L., Rodríguez, M.J. and Parets, J. (2002) 'Evolving hypermedia systems: a layered software architecture', *Journal of Software Maintenance and Evolution: Research and Practice*, John Wiley & Sons, Ltd., Vol. 14, No. 5, pp.389–405.

Gea, M., Medina, N., Rodríguez, M.L. and Rodríguez, M.J. (2004) 'Sc@ut: platform for communication in ubiquitous and adaptive environments applied for children with autism', *8th ERCIM Workshop. User Interfaces For All*, LNCS 3196, Springer, pp.50–67.

Halasz, F. and Schwartz, M. (1990) 'The Dexter hypertext reference model', *NIST Hypertext Standardization Workshop*, pp.95–133.

Kibby, M., Mayes, T. and Anderson, T. (1990) *Learning About Learning from Hypertext. Designing Hypermedia for Learning*, Berlin/London: Springer-Verlag, pp.227–250.

Koch, N. and Wirsing, M. (2002) 'The Munich reference model for adaptive hypermedia applications', *2nd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, LNCS 2347, Springer, pp.213–222.

Lowe, D. and Hall, W. (1999) *Hypermedia and the Web: An Engineering Approach*, Wiley.

Mayes, M. (1994) 'A method for evaluating the efficiency of presenting information in a hypermedia environment', *Computer in Education*, Vol. 18, No. 1, pp.179–182.

Medina, N., García, L., Torres, J.J. and Parets, J. (2002) 'Evolution in adaptative hypermedia systems', *Proceedings of the First International Workshop on Principles of Software Evolution*, IWPSE 2002, Orlando, USA, pp.34–38.

Medina, N., Molina, F. and García-Cabrera, L. (2005) 'Diversity of structures and adaptive methods on an evolutionary hypermedia system', *Journal IEE Proc.-Software*, Vol. 152, No. 3, pp.119–126.

Medina, N., Molina, F. and García-Cabrera, L. (2006) 'An adaptation meted by feedback in an evolutionary hypermedia system', *Proceedings on the Sixth International Conference on Web Engineering*, Palo Alto, California, pp.161–168.

Medina, N., Molina, F., García-Cabrera, L. and Parets, J. (2003) 'Personalized guided routes in an adaptive evolutionary hypermedia system', *EUROCAST 2003*, Lecture Notes in Computer Science 2809, Springer, ISBN: 3-540-20221-8, pp.196–207.

Molina, F., Medina, N. and García-Cabrera, L. (2006) 'An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems', *ACM International Conference Proceeding Series*, Workshop Proceedings of the Sixth International Conference on Web Engineering, Palo Alto, California, Vol. 155.

Novak, J.D. and Gowin, D.B. (1984) *Learning How to Learn*, Cambridge University Press.

Parets, J. and Torres, J.C. (1996) 'Software maintenance versus software evolution: an approach to software systems evolution', in *ECBS 1996*, pp.134–141.

Rivlin, E., Botafogo, R.A. and Shneiderman, B. (1994) 'Navigation in hyperspace: designing a structure-based toolbox', *Communications of the ACM (CACM)*, Vol. 37, No. 2, pp.87–96.

Schwabe, D., Rossi, G. and Barbosa, S. (1996) 'Systematic hypermedia application design with OOHDM', *Proceedings of the ACM International Conference on Hypertext (Hypertext'96)*, Washington, DC.

Stotts, P.D. and Furuta, R. (1989) 'Programmable browsing semantics in Trellis', *Hypertext'89 Conference*, pp.27–42.

Wang, W. and Rada, R. (1998) 'Structured hypertext with domain semantics', *ACM Trans. Information Systems*, Vol. 16, No. 4.

Knowl Inf Syst (2011) 29:629–656
DOI 10.1007/s10115-010-0357-1

REGULAR PAPER

Adaptation and user modeling in hypermedia learning environments using the SEM-HP model and the JSEM-HP tool

Nuria Medina-Medina · Fernando Molina-Ortiz · Lina García-Cabrera

Received: 14 January 2010 / Revised: 20 May 2010 / Accepted: 23 October 2010 /
Published online: 13 November 2010

Abstract This paper presents a model, called SEM-HP, which enables the development of evolutionary hypermedia systems that can adapt their functioning to each user and their structure to a particular group of users. The individual adaptation feature manages a user model that includes the personal data, experience, preferences, knowledge, and interests of each user. It applies knowledge-based adaptation techniques to the following tasks: (1) personalized selection of the navigation structure, (2) hiding and disabling of links to inappropriate information, (3) positive annotation of interesting links, (4) generation of guided routes, and (5) building of personalized conceptual summaries. The group adaptation feature uses transition matrices to model the navigational activities of a group of users and, based on these, suggests modifications to evolve the navigation structures defined by the author to bring them closer to the mental concept of the majority of users. The paper also introduces a general taxonomy of user models that makes it possible to classify any model according to various features of its structure and management. In addition, the taxonomy is used to classify the UM managed in SEM-HP, thus revealing its strengths and weaknesses. The last part of the paper describes a teaching experiment performed using the JSEM-HP tool, which is based on the SEM-HP model. This description includes the educational system created with the tool, the usage of the tool in the classroom, the evaluation performed after the tool was used, and the results obtained.

Keywords User model · Adaptation · Hypermedia system · Software evolution

N. Medina-Medina (✉) · F. Molina-Ortiz
University of Granada, Granada, Spain
e-mail: nmedina@ugr.es

F. Molina-Ortiz
e-mail: fmo@ugr.es

L. García-Cabrera
University of Jaén, Jaén, Spain
e-mail: lina@ujaen.es

1 Introduction

Hypermedia systems (HSs) structure and provide access to information through a network of multimedia nodes connected by links. These systems work as an associative memory, making them intuitive and popular. By going back to the origins of HSs, one can find that many of the characteristics of current HSs are inherited from the first pioneering systems. For example, the MEMEX system [1] defined in the forties the concept of a "trace" to enable nonlinear access to information, and in the sixties, the Xanadu system [2] implemented the concept of "transclusion" to include a document or part of a document into another by reference. Since then, diverse and numerous studies have been performed in this research area. Among them, the authors emphasize the widespread effort to define formal hypermedia models that support the construction of a particular HS from a software engineering perspective. Another major effort has been made in the direction of adjusting an HS according to the characteristics of each user.

Many different efforts have proliferated in the field of so-called adaptive hypermedia systems (AHSs), all with a common goal: to provide a customized way to reduce the problems of cognitive overload and disorientation that are typical of traditional HSs. Of course, to perform this adaptation, the system must handle certain information about the users, which can include their personal and social context, knowledge, experience, preferences, interests, goals, intentions, physical and psychological state, and movements. This set of data is stored, structured, and managed in what is known as the user model (UM). The UM is the image of the user inside the system; in fact, it has been defined informally as "the user's mirror" [3], and for the adaptation as performed to be fully effective, the UM must truly function as a mirror (that is, the UM must be continuously updated so that it always reflects the current state of the user). The system consults the UM to adjust its structure and functionality using previously defined adaptation mechanisms.

It is apparent from the scientific literature that the specific structure of the UM changes substantially depending on the adaptive system. For example, in the architecture AHA [4–6], it consists of a set of attribute-value pairs; the system described in [7] uses a graph-based representation of the user profile, which is an instance of the ODP ontology, and the Hyperbook KBS [8] uses a Bayesian network. As for the objectives of user modeling, many different versions can be found, including a general UM ontology for the interpretation of distributed UMs in intelligent semantic web environments, called GUMO [9]; a cognitive hybrid UM that combines attributes describing user intent with attributes describing an information retrieval system in a decision-theoretic framework [10]; a UM-based link annotation scheme to adapt an educational hypermedia system to the goals and knowledge of an individual user [11,12]; a method for learning a model from a set of Web logs augmented with the user's assessment of whether each page contains the information required, so further pages that might be useful can be identified [13]; a methodology that integrates textual content analysis to build an elaborate UM, which permits the personalization of a news service [14]; a user-modeling method for personalized selection of multimedia content, tested on a corpus of TV programs [15]; a three-dimensional UM (skill, knowledge, and urgency) used to generate cooperative responses to users in spoken dialogue systems [16]; and a system for filtering data on the Web, based on an adaptive data-filtering scheme in conjunction with a reevaluation approach to handle the diversity of Web data [17]. In addition, some efforts have been made to develop models and software tools that enable the design and construction of generic and reusable UMs. All agree that it is essential that the UM be a distinguishable component within the adaptive system. An example is **um** [18], a toolkit for building reusable, long-term user models based on evidence.

The current proposal, called SEM-HP, is a semantic, systemic, and evolutionary model that considers hypermedia adaptation on two levels: (a) adaptation to an individual user and (b) adaptation to a group of users. The first is an automatic adaptation that adjusts the functioning of the system to each user, and the latter is a structural adaptation that is based on the collective activity of a group of users and which, because of the scope of the structural changes, requires the collaboration of the author. SEM-HP is a formal model for the design, development, and maintenance of an AHS that fits particularly well with educational environments for several reasons: (a) it creates a semantic representation of the information, (b) it arranges the information into a conceptual structure, (c) it enables the visualization of this conceptual structure, (d) it provides direct navigation within the conceptual structure, (e) it calculates the extent of user knowledge as the information offered is browsed, (f) it provides a concept-based navigation mode, (g) it provides a navigation mode based on pedagogical prerequisites, and (h) it offers the possibility of requesting guided routes to achieve knowledge goals. All these features support a contextualized and personalized learning process that enables more efficient teaching and learning in an educational environment.

This paper is structured as follows: Sect. 2 outlines a taxonomy that makes it possible to classify any UM from twenty different perspectives. Section 3 explains the main elements of the model presented here, SEM-HP, with respect to the architecture, evolution, and navigation of the HSs being developed. Section 4 details the UM used in this context to perform individual and group adaptation. In addition, in this section, the UM used in SEM-HP is characterized according to the UM taxonomy outlined in Sect. 2. Section 5 describes a teaching experiment performed using the JSEM-HP software tool, which is based on the SEM-HP model. Finally, Sect. 6 briefly presents conclusions and recommendations for further work.

2 UM taxonomy

The importance of user modeling is unquestionable because it determines what corrections must be made to improve the user's interaction with the hypermedia information offered by the AHS. For this reason, in previous work [19], the authors developed a UM taxonomy that establishes a precise and complete set of classification criteria. This taxonomy makes it possible to situate the current UM, and any other, in relation to the various kinds of UMs that can be found in the AHSs developed so far.

The taxonomy contains twenty classification criteria organized into two groups: on the one hand, the criteria related to the structure and representation of the information stored in the model, and on the other hand, the criteria related to the management of that information. Two main aspects of information representation are considered: the features of the user stored in the UM and the way in which these features are structured. The key aspects of model management are three: initialization of the UM, update of the model once built, and use of the UM to adapt the model. According to this division, Table 1 summarizes the classification criteria concerning the structure of the UM and Table 2 the criteria for model management.

Of course, a UM can be hybrid with respect to any of these classification criteria. For example, most models contain long-term information, i.e., general user data that will remain stable over time (e.g., degree of experience in the use of HSs), and short-term information, i.e., highly specific user data that will change frequently (e.g., items of information visited in the HS).

Table 1 UM taxonomy—structure

Classification criteria		Types of UMs		
Granularity		Fine-grained (individual UM)		
		Coarse-grained (group UM)		Several partial groups
				One group with all users
User representation		Stereotypes		Social stereotypes
				Knowledge stereotypes
				etc.
		Overlay model		
Main trace (or traces)		Interests and preferences (recommendation)		
		Knowledge (learning)		
		Location (movement)		
		Tasks (activity)		
		etc.		
Structuring	Levels of structure	Flat structure		
		Episodic structure		
	Formality	Informal structuring (attribute-value pairs)		
		Formal structuring		Bayesian networks
				Petri net
				etc.
Generality		Short-term (specific information)		
		Long-term (general information)		
Conceptualization		Nonconceptual		
		Non-conceptual model	Basis	Based on diagrams of objects
				Based on semantic net
				Based on ontologies
				etc.
			Domain	Domain-dependent
				Domain-independent
Psychological aspects		Noncognitive		
		Non-cognitive model		Perception
				Memory
				Emotion
				etc.
Monitoring		No monitoring		
		User interface events		
		Localization and/or movement		
		Audio or video		
		etc.		
Context		Context-independent		
		Context-dependent		Technological context
				Institutional context
				etc.

Table 2 UM taxonomy—management

Classification criteria	Types of UMs	
Initialization	Null	
	Partial or total	By default
		By forms
Update	Static (rarely or never updated)	
	Dynamic	Explicit or active
		Implicit or nonintrusive
Prediction	No prediction	
	Predictive model	Content based model
		Collaborative model
Statistical analysis	No statistical analysis	
	Statistic model	Linear model
		Term frequency–inverse document frequency (TFIDF)-based model
		Markov model
		Neural networks
		etc.
Stability	Short-term (volatile information)	
	Long-term (permanent information)	
Inference method to adaptation	Theory-based inference method	
	Data-based inference method	
Direction of the inference to adaptation	Push UM (final values are stored)	
	Pull UM (values are processed for each adaptation)	
Direction of the adaptation	Exploitation of the UM (convergent adaptation)	
	Exploration from the UM (divergent adaptation)	
Visibility	Open UM (glass box)	
	Closed UM (black box)	
Portability	Non-standard	
	Portable	Learner information package (LIP) standard
		Personal and Private Information (PAPI) standard
		etc.
Centralization	Centralized	
	Decentralized	

3 SEM-HP model

SEM-HP [20] is a SEMantic, systemic, and evolutionary model for the development of adaptive HyPermedia systems. It is called "semantic" because it is based on semantic representations of hypermedia, such as semantic networks; it is called "systemic" because, based on the principle of separation of concerns, it conceptualizes the AHS as four subsystems; and it is called "evolutionary" because it takes care of the evolution of the system, guaranteeing that the system remains consistent when it changes.

SEM-HP supports both the development and evolution of AHSs because over the whole life cycle, the evolution of the system is considered as part of the construction process. To accomplish this, SEM-HP's architecture, shown in Fig. 1, is structured in two dimensions: the vertical dimension, in which the system is structured as four interrelated subsystems (memorization, presentation, navigation, and adaptation, which will be further detailed below), and the horizontal dimension, in which two abstraction levels are distinguished: the system layer and the meta-system layer.

The system layer stores all the logical models built during the development process of each subsystem and is the AHS that is used by the final users, while the meta-system layer contains the mechanisms needed to perform consistently the maintenance of the models included in each subsystem and is embodied in the evolutionary software tool used by the author to create and modify the AHS. To do this, the meta-system implements evolutionary mechanisms that check the integrity of each change that the author tries to perform on the AHS. The meta-system rejects the change if it is inappropriate, but if the change is accepted, the meta-system performs the change and propagates the necessary modifications so that the system stays within predefined integrity restrictions.

3.1 SEM-HP architecture

The SEM-HP separates the aspects of memorization, presentation, navigation, and learning in four interrelated and interacting subsystems (vertical division in Fig. 1).

The **memorization subsystem** stores the knowledge and information domains of the AHS, which are represented by means of a conceptual structure (CS). The CS is a semantic network with two kinds of nodes: concepts and items. Concepts are semantically labeled ideas that are linked by means of conceptual associations. Items are pieces of information offered by the AHS, for example, an html page or a pdf document. Each item is linked with one or more concepts by means of a functional association, which is labeled by specifying the function (or role) of the information contained in the item with regard to its associated concept, for example, introduction, definition, example, etc. Graphically, a CS has the appearance shown in Fig. 2. Concepts (such as "Class") are represented by ovals and items (for example, I1) by rectangles.

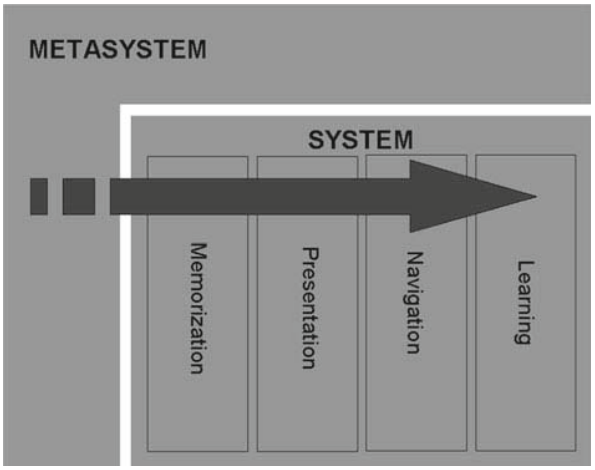


Fig. 1 SEM-HP architecture

The **presentation subsystem** characterizes the different views created by the author from the conceptual structure of memorization (CS_M). Each view is represented by a conceptual structure of presentation (CS_P), which consists of a subset of the concepts, items, and associations included in the CS_M, possibly in a different spatial arrangement. The author labels each CS_P with the knowledge subdomain, or subdomains it captures, and the degree to which it captures them. For example, CS_P^1 (Fig. 3) would be labeled with the subdomain "Class" with a degree of 100%.

The **navigation subsystem** establishes for each CS_P a set of ordering rules (Ro), which defines a partial order for access to the items in that presentation and which is based on the conceptual relations included in it. For each item, an Ro is created by default for every conceptual relation that can be navigated to the concept associated with the item. Therefore, in CS_P^1 (Fig. 3), there would be two Ro for item I15 (introduction to "MetaClass"): $Ro^1(15)$ for the conceptual relation "has" and $Ro^2(I15)$ for the conceptual relation "is a". Each default rule requires an immediately *previous* visit to one of the items associated with the origin concept of the conceptual relation. Unless the author defines it differently, a conceptual relation is navigable only from its origin concept to its destination concept. In addition, the author has methods to modify an Ro by requiring another item of the CS_P to have been visited *before* the immediately previous step (two or more navigational steps before the item). An author wishing to require previous visits to two or more items should join "before" predicates by "and" operators. An author wishing to state that it is enough to visit any one of these items will join "before" predicates by "or" operators. For example, to visit I15 through the relation "has", which starts at the concept "Class", the Ro as modified by the author could be:

$$\mathbf{Ro}^1(\mathbf{I15}) : [\mathbf{previous}(\mathbf{I5})\ \mathbf{or}\ \mathbf{previous}(\mathbf{I30})]\ \text{and before(I6)} \tag{1}$$

The union of a set of Ro and the CS_P in which it has been defined is called a conceptual structure for navigation (CS_N). Following an incremental approach, from the same CS_P, the

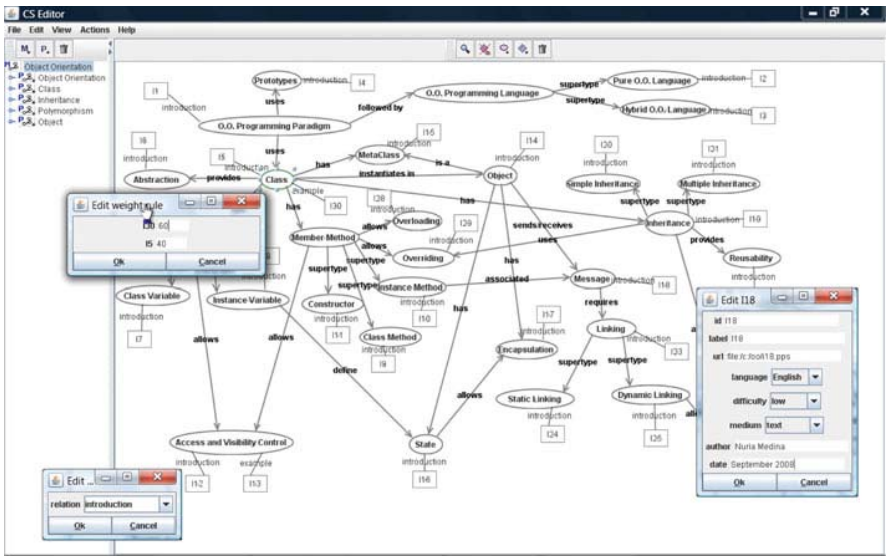


Fig. 2 CS of the memorization subsystem in the object-oriented paradigm (CS_M)

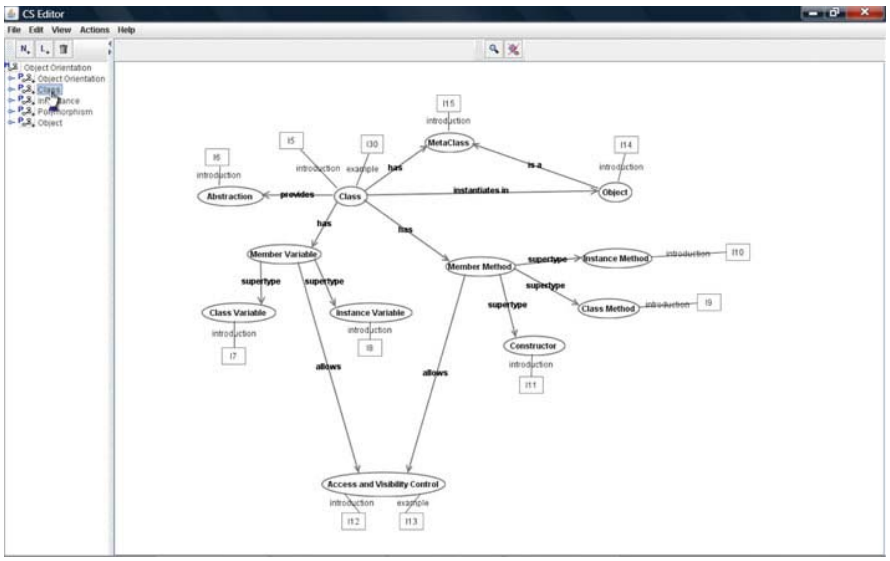


Fig. 3 CS of the presentation subsystem of the concept "Class" (CS_P^1)

author can define several sets of Ro, thereby defining several CS_N, each one with a different navigation criterion.

The **learning subsystem** takes care of user adaptation. An important part of the adaptation as performed is related to the user's knowledge and the items' learning requirements. For this reason, SEM-HP is especially well-suited for creating educational AHSs. Knowledge is quantified using five semantic labels: "null", "low", "medium", "high", and "total". A set of update rules is used to model the user's knowledge acquisition process as he accesses the information. A set of knowledge rules establishes the pedagogical prerequisites existing between the items.

For every item in the CS_N, there is one *update rule* (Ru) that by default gives the user total knowledge of the item read. Nevertheless, the author has methods to modify the Ru to indicate that reading an item also provides knowledge about other related items (uniting all update predicates by "and" operators) or to modify the kind of update performed to the visited item's knowledge. An update is defined by a logic predicate and can be: (a) fixed at a specific knowledge degree or a fixed increment above the current knowledge level, (b) absolute if it implies a specific knowledge level or relative if it refers to the previous knowledge level about the visited item, and (c) performed every time the item is visited or only the first time. The available update predicates are: Fix-abs, Inc-abs, Fix-rel, Inc-rel, Fix-abs$_{first}$, Inc-abs$_{first}$, Fix-rel$_{first}$, and Inc-rel$_{first}$. For example, Eq. 2 shows an Ru which specifies that, after visiting I13, the user's knowledge about I13 (example of "Access and Visibility Control") is set to "total" and that the first time I13 is visited, the user's knowledge about I12 (introduction of "Access and Visibility Control") is increased by one level. A visit to an item can mean simply reading the item or may impose additional requirements such as a minimum reading time or passing an associated comprehension test.

$$\textbf{Ru}(\textbf{I13}) : [\textbf{Fix-abs}(\textbf{I13}, \text{“}\textbf{total}''\text{)}] \text{ and Inc-abs}_{\text{first}}(\text{I12,1+}) \tag{2}$$

For every item in the CS_N, the author has methods to define as many *knowledge rules* (Rk) as possible alternative ways to prepare to learn the item. Each Rk defines the minimum knowledge level that the user must have to understand the information in the item (**accessibility restrictions**, operators: $\geq$, $>$) and the maximum recommended knowledge level above which reading the item would not provide the user with new information (**idoneity restrictions**, operators: $\leq$, $<$, $=$). Accessibility restrictions are combined using "and" and "or" operators to form the accessibility predicate. Similarly, idoneity restrictions are joined to form the idoneity predicate. Both predicates are joined with the "and" operator. In this way, if there is no Rk associated with an item, this means that reading the item is always allowed and always recommended. For example, an Rk saying that to access I12 (introduction of "Access and Visibility Control"), it is necessary to have at least a "medium" knowledge degree about the introduction of Instance Variable (I8) or Instance Method (I10), and that for a visit to I12 to be considered idoneous, it is appropriate not to know perfectly the example of "Access and Visibility Control" (I13), would be:

$$\mathrm{Rk}^{\mathrm{i}}(\mathrm{I12}) : [\mathrm{K}(I8) \geq \text{“medium”} \text{ or } \mathrm{K}(\mathrm{I10}) \geq \text{“medium”}] \text{ and } [\mathrm{K}(\mathrm{I13}) \leq \text{“high”}], \quad (3)$$

where K(I) is the level of knowledge about I.

The set of rules Ru $\cup$ Rk must allow all items to be reached. This makes it possible to construct a reachability tree from this set of rules. Figure 4 shows the first level of the tree generated for the CS_M of Fig. 2. The root of the tree represents a knowledge level of "null" about the 30 items in the example. The rules used can be seen on the figure itself. According to these rules, initially only I1 can be visited. Once I1 has been visited, items I4 and I5 are made accessible. The nodes of the tree grow in knowledge as new items are visited, and the depth of the tree increases.

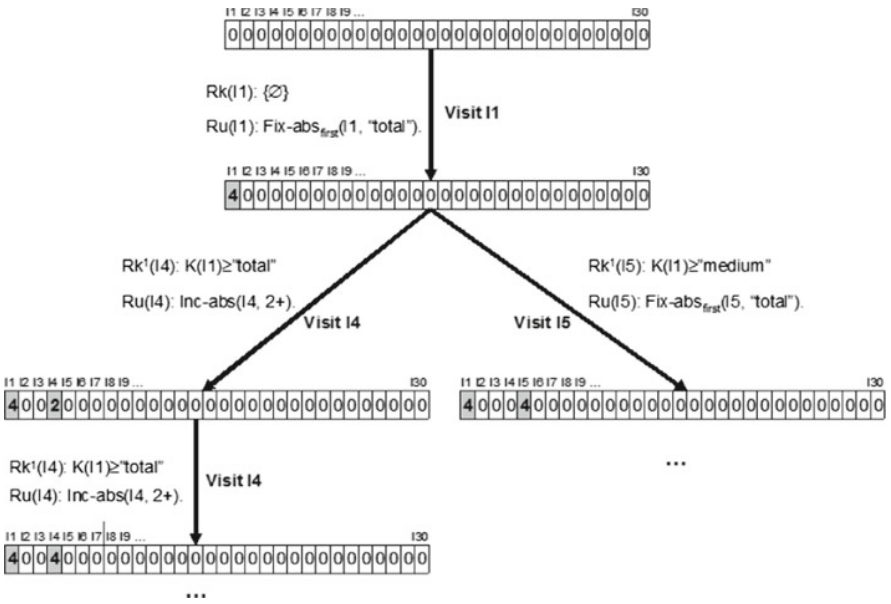


Fig. 4 Reachability tree

Again, for each CS_N, different sets of Ru∪Rk can be defined, thereby specifying different conceptual structures of learning (CS_L), with different adaptation schemes. In addition, the author labels each CS_L with degrees of experience in navigation and in the subject matter for which this CS_L has been designed. For convenience to the author, the level of experience can be expressed using the stereotypical terms of "novice", "beginner", "intermediate", and "expert".

3.2 Evolution of the AHS

Each one of the four subsystems has a meta-system that takes care of its evolution [20]. The meta-system contains a set of evolutionary actions (ACe) that are used by the author to modify the system. Each ACe contains a set of pre- and post-conditions that must hold if the change is to be carried out. The meta-system automatically propagates the changes in the affected subsystem, ensuring its internal consistency after the change. Changes performed within a subsystem could also affect other subsystems; for example, some changes in the memorization subsystem might affect the presentation subsystem, and some changes in the presentation subsystem might affect the navigation and learning subsystems. To ensure overall consistency, the meta-system also performs external change propagation. For example, if I13 is removed from the CS_M shown in Fig. 2, it is also removed from the CS_P^1 (Fig. 3), its associated Ro and Ru are removed (for example, Ru(I13) in Eq. 2), and the predicates defined for I13 in other rules are removed (for example, the idoneity predicate K(I13) $\leq$ "high" is removed from Rk^i(I12) in Eq. 3).

3.3 Navigation of the AHS

In summary, an AHS developed with SEM-HP is composed of a CS_M that captures the overall knowledge domain and several CS_L, each of which has a CS_P, a set of Ro, and a set of Rk and Ru. The author labels each CS_L with the subdomains of knowledge that it captures and the level of experience required for browsing it, and then, according to the labels of the CS_L, the system chooses the most appropriate CS_L for the user depending on his previous experience and the subdomain of interest. In addition, four different modes are offered for navigating the chosen CS_L [21]: **traditional**, **by concepts**, **restricted by conceptual relations**, and **restricted by knowledge**. In these four modes, the user's knowledge acquisition process while he browses the system is calculated in the same way: when the user visits an item, the item's Ru will be executed only if the user's knowledge state satisfies the accessibility restrictions of any of the item's Rk, or if the item has no associated Rk.

In **traditional navigation**, such as happens on the Web, when the user clicks on an item, its contents are shown. The difference is that, in the present case, the user selects the items directly on the graphical representation of the CS_P (in other words, a semantic net is browsed). In this way, the user visualizes the conceptual domain that underlies the requested information. **Navigation by concepts** also has no access restrictions, but only the concepts and conceptual associations of the CS_P are shown, and when the user selects a concept, he is given a personalized summary consisting of the items of information associated with the concept. In **navigation restricted by conceptual relations** and in **navigation restricted by knowledge**, the information domain (items) of the CS_P is usually partially shown; the system hides and disables the items for which the user does not satisfy the restrictions imposed by the Ro (restricted by conceptual relations) or the accessibility restrictions imposed by the Rk (restricted by knowledge).

Of course, the latter type of navigation is more appropriate in educational environments where the student's learning process is to be directed according to a set of pedagogical prerequisites previously set by the tutor. Therefore, at a given moment, the student has available only those items that contain information that he is prepared to understand. As the user explores new items and acquires new knowledge, he is then able to learn other, more complex items. Consequently, the adaptation makes these items accessible in the navigation structure, increasing the domain of information offered.

4 The SEM-HP UM

In SEM-HP, there is an individual UM for each user and a group UM for the set of all users. Both the individual UM and the group UM are structured as learning episodes. In the first case, each episode corresponds to a CS_M accessed by the user, while in the second case, there is one episode for each CS_P offered in the HS.

At a second structural level, each episode in the individual UM is represented by mean attribute-value tuples. Each tuple stores information about an item or concept included in the corresponding CS_M. The tuple therefore specifies an overlay model defined on the knowledge domain captured in that episode. The model includes short-term information, such as the specific degree of user knowledge about every information item and concept. In addition, the UM stores other general information, such as the user's level of experience, which is represented by means of stereotypes.

In the group UM, each episode is represented using a transition matrix. Each matrix is obtained from the navigation performed by users on the corresponding CS_P. This makes it possible to infer a mental model of the knowledge subdomain captured in the CS_P, which all users share. To do this, the transition matrix is analyzed statistically.

The individual UM is open and dynamic; therefore, it allows both explicit and implicit updating. The group UM is also dynamic, but open only to the author, who receives suggestions for design changes in the conceptual structures according to the collective navigation of users as captured in the transition matrices. In both models, to carry out the adaptation, a data-based inference is made. The data collected are related primarily to the items that the user visits (for simplicity, this is considered to be the number of times that the item is viewed, but it can also include reading time, the result of a comprehension test, etc.). These data are used according to a push approach to update automatically the group UM (using transition matrices) and the individual UMs (using Ru) with their final values, that is, the values that will be directly used for adaptation.

Thus, the union of the two models is a novel integration of a set of appropriate techniques for creating an appropriate adaptive learning environment. Next, Sect. 4.1 explains in more detail the individual UMs, and Sect. 4.2 outlines the most important concepts of the group UM.

4.1 Individual UM

The individual UM is the internal representation of each user that is kept by the system to enable it to adapt its functioning to the user's specific features and needs. To do this, the system always maintains a representation of the user's current state. This state represents not only the user's acquired knowledge, but also his interests, preferences, and previous experiences. Table 3 specifies the complete set of attributes stored in the UM, grouped into five categories: personal data, knowledge, experience, preferences, and interests. Figure 5 shows

Table 3 Attributes of the individual UM

Category	Attributes
Personal data	Access password, name, surname, gender, age, and occupation
Knowledge	Number of visits and degree of knowledge
Experience	Experience with the subject matter and experience in navigation
Preferences	Preferences about items, preference for short guided routes, and preferences for summary structure
Interests	Knowledge subdomains, interesting items, interesting concepts, and knowledge goals

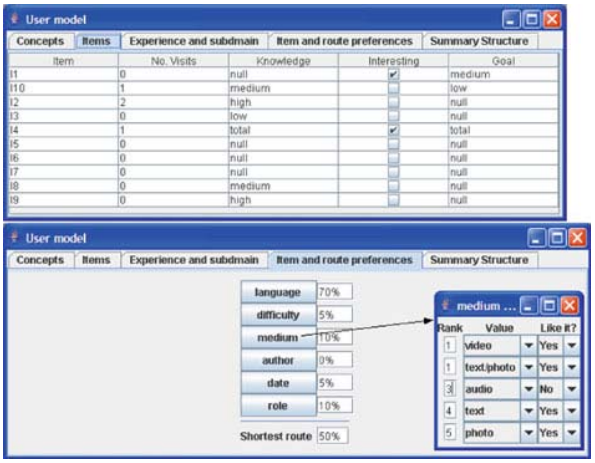


Fig. 5 User model: tabs for items and route preferences

part of one instance of the UM. Sections 4.1.1 to 4.1.5 detail the attributes for each category with respect to their initialization, update, and usefulness for adaptation.

4.1.1 Personal data

Personal data are used to customize the communication process, for example, to address the user by name or to modify the presentation style according to his age and occupation. These attributes are also useful to restrict the degree of control that the user has over his user model, establishing rules like: "a *student* less than *18 years old* cannot change the knowledge state that the system has inferred for him." Personal data are updated only by explicit user request and, due to their administrative nature, are treated differently from other types of data in the UM.

4.1.2 Knowledge

This category considers the number of visits performed and the degree of knowledge attained about every item and concept in the CS_M (Fig. 5). Unless the user sets it differently, the number of visits is initialized to zero and the knowledge degree to "null".

The number of **visits** to an item is increased by one when the user accesses the item's content, even when this content is part of the summary of a concept. In the same way, the number of visits to a concept will be increased every time the user visits any of its associated items or obtains a summary of the concept. The **degree of knowledge** about the items is updated by the execution of update rules, as explained in Sect. 3.1. The degree of knowledge about the concepts is obtained from the knowledge that the user has about each of its associated items, which is given a weight depending on the relevance that the item has to learning the concept. Therefore, the system defines for each concept a weight rule (Rw) with a default uniform weight distribution that can be changed by the author. For example in the CS_M shown in Fig. 2, Rw(Class) could be set to: K(Class)=40% K(I5)+ 60% K(I30).

The number of visits and the degree of knowledge is annotated on each item and concept in the navigation structure provided to the user. This makes the user aware of his process of navigation and learning. In addition, user knowledge about the items is used to evaluate the Rk and to determine whether, given the user's current knowledge state, an item is accessible and idoneous. Specifically, if this state satisfies all the accessibility restrictions of any of the item's Rk, the item is accessible, and if the knowledge state also satisfies the idoneity restrictions, the item is idoneous. During knowledge-restricted navigation, inaccessible items are hidden and disabled, and accessible but non-idoneous items are negatively annotated to discourage visits to them [21].

4.1.3 Experience

Experience in the subject matter reflects the general knowledge that the user has about the conceptual domain of the AHS. If it is not explicitly initialized, this value is set to "novice" (value 0) by default, although the user can update it manually at any time or can request this value to be estimated by the system by calculating the average user knowledge about all the concepts in the CS_M. **Experience in navigation** represents the user's skill in hypermedia navigation. It can also be automatically updated by a partial estimate obtained from the number of visits performed by the user in the system. Both experience attributes are used to choose, among all the available CS_L, the one that best suits the user (as mentioned before, each CS_L is labeled with the subdomains it captures and the experience in the subject matter and in navigation that its users should have).

4.1.4 Preferences

With respect to items, the user can specify his preferences regarding six features: **language** ("English", "Spanish",. . .)**, author, date of creation** (exact date or date range)**, level of difficulty** ("high", "medium",. . .), **role** ("example", "definition",. . .), and **media** ("audio", "video", …). For each feature, the user defines an ordered list of values indicating which of them he *prefers* and which of them he does *not prefer*. For example, the user may specify that he prefers items that contain videos or photos with explanatory text, in this order, and that he does not like those that contain only audio. In addition, the user assigns a weight to each feature depending on how important it is for him. For example, the language in which the item is written can be considered fundamental (high weight), and the item's author can be considered irrelevant (low or zero weight) (Fig. 5). These preferences are taken into account when guided routes are generated [22]. A guided route is a set of items and a strict order for visiting them, which will enable the user to reach a desired knowledge state. The system

builds the guided route by searching the reachability tree built from the set of Rk and Ru (Fig. 4). Each node in the tree represents a possible knowledge state composed by the degree of knowledge about each item in the CS_M. An arc going from one node to another represents a visit to an item and the execution of its Ru about the knowledge state as represented in the origin node. The starting node in the search tree represents the user's current knowledge state, and the final node must satisfy the desired knowledge goal. Each arc is assigned a cost, which is calculated by comparing the features of the item to which a visit is implied by the arc with the preferences specified by the user. In this way, the route proposed to the user is the one that has the lowest cost.

The attribute **preference for short routes** refers to the length of guided routes and reflects the importance that the user gives to routes that have the least possible number of items. The user establishes a weight between 0 and 100 (100 means that the user wants the shortest possible route regardless of its items' features).

The **structure of the summaries** provided in navigation by concepts is initially defined by the author. To do this, he establishes an ordered list of item roles, indicating which of them are *compulsory* and which are *optional* in a conceptual summary. As a result, in a conceptual summary, there is a frame for each role in a defined order. Frames corresponding to optional roles are collapsed, and compulsory frames show the content of the items that are associated with the concept through the corresponding role. To obtain personalized summaries, the user can specify his preferences in the UM, or he can generate a summary directly by reordering, expanding, and collapsing frames in the summary obtained [21].

4.1.5 Interests

The user indicates the **knowledge subdomain** in which he is most interested. So, in the process of selecting the CS_L, the system takes into account the degree to which that subdomain is captured by each CS_L. However, for a given knowledge subdomain, the user may be more interested in learning about some items than in learning about others. Because of this, he is allowed to mark as **interesting** the items he prefers (Fig. 5). In addition, the system can infer interesting items based on the user's preferences, establishing as interesting those items for which the features match sufficiently well the user's preferences. Interesting items are positively annotated during navigation restricted by knowledge. The user can also establish a **knowledge goal**, specifying not only the items in which he is interested, but also the minimum degree of knowledge that he wants to achieve about each one of them (Fig. 5). The knowledge goal can be automatically deduced from the set of interesting items, requiring "total" knowledge of them. In any case, the goal, once established, is used to generate a guided route that leads the user to fulfill the goal.

4.2 Group UM

The group UM focuses only on modeling the navigation of the user group, so a more accurate name for it would be "group navigation UM". In practical terms, it aims to capture the mental concept that the users have of the provided navigation structures [23]. To do this, the group model is represented as a set of transition matrices. Specifically, there is a transition matrix for each CS_P (called TM_P) and a transition matrix for the CS_M (called TM_M). Each TM_P matrix has a row and a column for each concept included in the corresponding CS_P (Fig. 6 shows the structure of the TM_P^1 created for the CS_P^1 in Fig. 3). The intersection between a

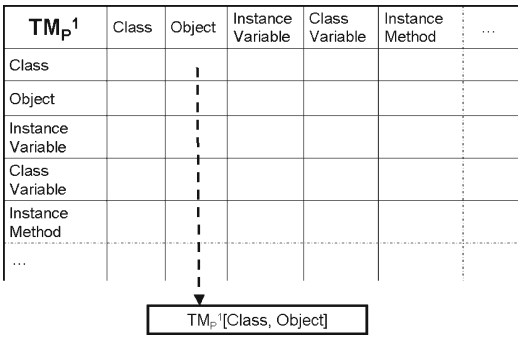


Fig. 6 TM_P^1: structure of the transition matrix for CS_P^1

row and a column will contain the number of times that a user has gone from the concept labeling the row to the concept labeling the column. Initially, each cell TM_P [*ci, cj*] is set to zero, and its value is increased by one every time that a user browsing a CS_L based on that CS_P selects an item associated with the concept *cj* when his last-visited item was associated with the concept *ci*. The TM_M matrix is built by combining the matrices associated with its presentations. In this way, information can be obtained both on the complete structure and on each of the partial structures created from it.

To obtain the matrices, only the traditional navigation mode is considered, because in this mode, users can select items freely, going from one concept to another according to their own conceptual schemata. Afterward, the system compares the conceptual schema implied by the navigation behavior of the group of users with the conceptual structures created by the author, obtaining: (a) meaningful features of the most visited and least visited CS_P, such as the subdomain of knowledge that they capture or the degree of experience needed to use them, (b) concepts and pairs of concepts (that is, impossible conceptual relations) that have not been included in any CS_P, and (c) differences between the real structures as defined by the author and the virtual structures as defined by the group of users whose behavior is captured by the transition matrices. The system analyzes these differences and suggests to the author: (1) the addition of conceptual relations that are not present in the real structure, but have been conceptualized by the group of users (if the users go significantly often from one concept to another, it can be assumed that they conceive of an association between them); and (2) the removal of existing conceptual relations that are infrequently used by the users. The author can accept or reject these structural changes; if he accepts them, they will be carried out through ACe functions that guarantee the system's consistency. This adaptation (evolution) of the conceptual structure enables future users to benefit from previous users' navigation strategies.

4.3 Characterization of the SEM-HP UM using the taxonomy

To complete the characterization of the UM used in SEM-HP, it will now be described with respect to the criteria set out in the taxonomy presented in Sect. 2. As shown in Table 4, the only aspects that are not covered by the SEM-HP UM are the inclusion of contextual and psychological information about the user and the possibility of exporting the UM created in a SEM-HP AHS to other AHS, which were not created according to the SEM-HP model.

Table 4 SEM-HP UM

	Classification criteria	SEM-HP UM
UM structure	Granularity	Hybrid: Fine-grained to model each individual user and coarse-grained to model the whole set of users
	User representation	Hybrid: Stereotypes for the user's experience and overlay model for the user's knowledge, interests, and goals about the conceptual and information domains
	Main trace	User's knowledge (in addition, interests and preferences are also traced)
	Structure	Hybrid: For the individual UM, a learning episode is defined for each CS_M, and attribute-value tuples are used to instantiate each episode. In the tuples, the attributes refer to concepts and items of information, and the values indicate degrees of knowledge, experience, interest, or preference about them
		For the group UM, a learning episode is defined for each CS_P, and a transition matrix is used to represent the navigation performed by the users over that CS_P
	Generality	Hybrid: The model includes both general or long-term information, such as level of experience with the subject matter or the subdomain of interest, and specific or short-term information, such as the specific degree of knowledge about every information item
	Conceptualization	The conceptual structure is an explicit representation of the concepts and conceptual relations by means of a semantic net which depends on the knowledge domain
	Psychological aspects	No
	Monitoring	Monitoring of the items selected by the user to obtain the number of visits and degree of knowledge about the items
	Context	No
UM management	Initialization	Total initialization: all the information has a default initialization (predefined by the author) that is applied if the user does not explicitly initialize it, except, of course, for personal data which must be initialized by the user. Explicit initialization is especially advisable for level of experience, subdomain of interest, and preferences
	Update	The UM is dynamic and allows both explicit and implicit updates: explicit because the user can directly modify the model's contents, and implicit because the system automatically obtains certain information such as the user's knowledge, items visited, interesting items from preferences, etc.
	Prediction	A collaborative prediction model is used when assuming that the navigation performed by the current set of users will be adequate for a future user. Because of this, the real conceptual structure is adapted based on the virtual conceptual structure defined by the group of users during their navigation and captured in the transition matrices
	Statistical analysis	A statistical analysis is performed on the transition matrices to obtain the most /least used presentations, the most /least visited concepts, and the most /least followed conceptual relations
	Stability	Hybrid: the model includes both stable or long-term information such as the level of experience in the subject matter and changeable or short-term information such as the number of visits to the information items. Other information, such as interests and preferences, can be long term or short term depending on the user

Table 4 continued

Classification criteria	SEM-HP UM
Direction of inference	Push model based on weight and update rules (Ru + Rw)
Inference method	Data-based method to collect individual usage data such as items visited, conceptual transitions, or the degree of knowledge attained about them (in some way, the group UM can be considered as a model predefined through past interactions, which makes it possible to improve future user interactions)
Direction of adaptation	Exploitation in restricted navigation; exploration in free navigation
Visibility	Open or glass-box UM, which allows the user to change the UM contents directly
Portability	No
Centralization	Centralized

5 A teaching experiment using JSEM-HP

JSEM-HP [24] is a tool based on the SEM-HP model that implements both author and user interaction. On the one hand, the JSEM-HP tool provides the author with the mechanisms to create the AHS and to manage its later maintenance, and, on the other hand, it provides the user with the interface to navigate the created AHS, applying the corresponding adaptation depending on the updated contents of the UM.

To assess SEM-HP's teaching usefulness, a group of teachers at the University of Granada has created, using the JSEM-HP tool, an educational AHS for the subject "Object-Oriented Programming (OOP)" that is being taught (in Spanish) at the Technical College of Informatics and Telecommunication Engineering at the university. The basic aim of this system is to provide the students with an overall view of the content that is taught in the classroom. In other words, a student who navigates the system completely must have obtained an introductory view of the main concepts of OOP. Specifically, the CS_M of the AHS as created (Fig. 2) contains 30 concepts related to object orientation ("Object-Oriented Programming Paradigm", "Class", "Member Variable", "Member Method", "Overriding", "Object", "Message", "State", "Encapsulation", "Inheritance", "Reusability", "Polymorphism", "Linking", etc.) and as many conceptual relations among them (Class "instantiates in" Object, Member Method "allows" Overriding, Inheritance "provides" Reusability, etc.). In this case, associated with every concept, there is one document containing a slide that presents a brief introduction to the associated concept. Therefore, the total number of information items is 30.

To avoid comprehension problems due to the large number of concepts in the CS_M, the conceptual domain was divided into four smaller presentations: one for the "Class" conceptual subdomain (Fig. 3), another for the "Object" conceptual subdomain (Fig. 7a), another for the "Inheritance" conceptual subdomain (Fig. 7b), and another for the "Polymorphism" conceptual subdomain (Fig. 7c). Each of these presentations contains approximately ten concepts, which can be easily visualized at a glance.

At the learning level, knowledge rules that define pedagogical prerequisites were established to ensure that when a student reads any information item, he or she is prepared to understand it. In other words, the student must have the previous knowledge that is required to be able to learn the concept with which that information is associated. For example, it was established that to understand without difficulties item I16 that introduces the concept

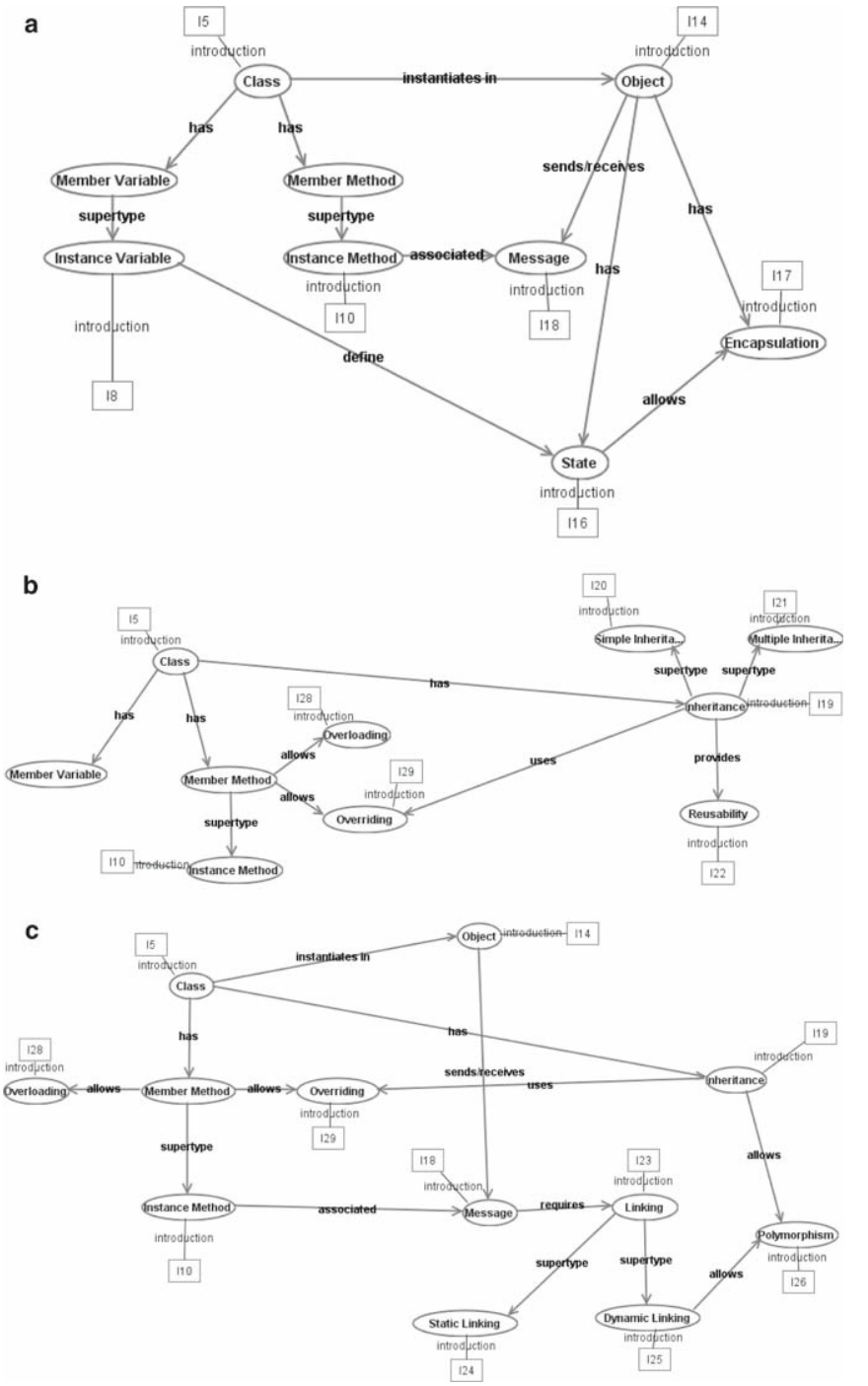


Fig. 7 **a** CS_P^2 for "Object", **b** CS_P^3 for "Inheritance", **c** CS_P^4 for "Polymorphism"

of "State", the student must have previously learned the notion of "Object" introduced in I14 and the concept of "Instance Variable" introduced in I8 (the formal knowledge rule would be: Rk(I16): [K(I14) ≥ "high" and K(I8) ≥ "high"]). The default update rules were not modified. In this way, every time the user studies an item, the knowledge degree about the item is set to "total" (the formal update rules are the following: ∀i Ru(Ii): Fix-abs(Ii, "total")). The level of knowledge about the item read is updated to "total" because the level of difficulty of each item is carefully set so that any student of the subject is able to understand the item with a single reading. It has been taken into account that the students are third-year college students who already know at least one programming language. Because the Rk ensure that students always have the necessary pedagogical prerequisites, when a student reads the introduction associated with a concept, it is perfectly understood. All the items contain a very brief and simple definition. For example, the slide that is visualized in full-screen mode when the student selects item I5 associated with the "Class" concept contains only this text: "The class is the unit of programming in object-oriented languages, that is, the programmer works by writing classes. A class defines the common behavior and structure of a set of objects."

During a 1-h lecture, the students navigated the four presentations in the same order in which they have been presented in this paper. Although the tool does not force the user to follow a given order for opening the presentations or to finish each presentation before opening another one, the teacher guided the navigation process so that first the Class presentation was finished, then the Object and Inheritance presentations, and finally the Polymorphism presentation. Because the purpose of the system was educational, the navigation mode used by the students was obviously the knowledge-restricted navigation mode. Therefore, in each presentation, only those items that the student was prepared to understand were accessible, that is, those items for which the current knowledge state of the user satisfied the pedagogical prerequisites expressed by knowledge rules. The remaining items were hidden and disabled, so that the students were not allowed to access information that was still "forbidden" to them. Specifically, when the presentation for the Class conceptual subdomain (Fig. 3) was initially opened, the student could access only item I5, which contains the introduction to the concept of "Class". Once this information had been read, items I6, I8, I10, I11, I14, and I30 became accessible. These items contain introductory definitions of the concepts of "Abstraction", "Instance variable", "Instance method", "Constructor", and "Object", as well as one example of "Class". The student can visit the accessible items in any desired order, and this navigation determines the sequence in which new items become accessible, until every item in the presentation can be accessed. Of course, when an item becomes accessible in a presentation, it remains accessible in all the presentations in which it appears. In this way, once the Class presentation had been completed, when opening the Object presentation, the only items that had not been previously visited were I16 (introduction of the concept of "State"), I18 (introduction of the concept of "Message"), and I17 (introduction of the concept of "Encapsulation"). Items I16 and I18 are initially accessible, while I17 is hidden and disabled because it requires the other two to be properly understood.

Figure 8 shows the navigation sequence just described. Already-known items are shown with a black background because the tool uses a grayscale code to show directly in the tool the user's level of knowledge about every item or concept: white means "null" knowledge, black means "total" knowledge, and different shades of gray from lighter to darker mean "low", "medium", or "high" knowledge. The text and border of the accessible items is black, while the inaccessible item, I17, is shown in light gray, which is close to imperceptible. Although students can visualize their learning processes directly in the conceptual structure using the grayscale code just mentioned, the contents of the UM can also be explicitly seen by choosing the "user model" option in the top toolbar.

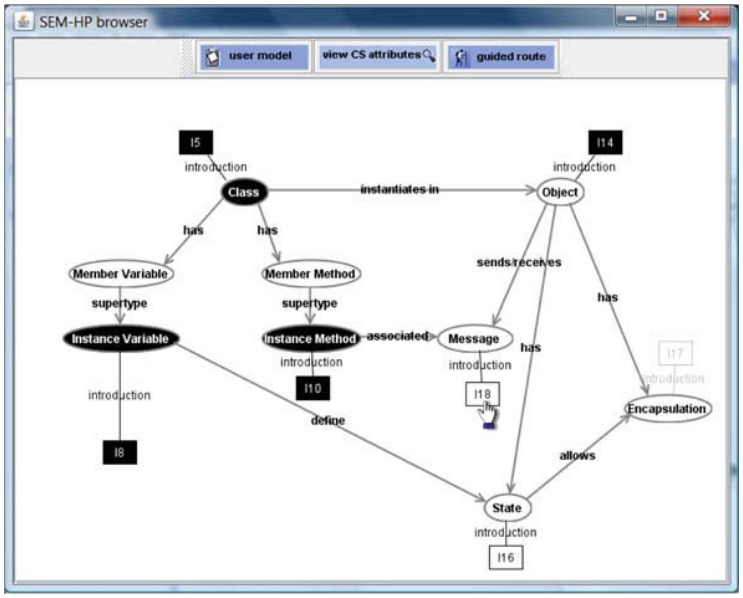


Fig. 8 Browsing CS_P for "Object"

After the experiment, a double evaluation was performed: on the one hand, the students' learning was succinctly assessed, and on the other hand, their usage of the tool was evaluated. For the first assessment, in later lectures, the teacher posed systematic questions when one of the concepts that had been presented using the AHS appeared, to check whether the students remembered the general concept that had been transmitted to them. After this subjective evaluation, the teacher concluded that the students had an overall view of the subject that enabled them to follow the lectures better than in previous years.

For the second evaluation, the same day that the tool was used, a questionnaire consisting of 36 questions was given to the students (the questions included in the survey can be seen in the "Appendix"). The survey about the tool was voluntarily and anonymously completed by 31 students aged 20–25 years. The questionnaire took about 30 min, and the questions were divided into three blocks. The first block consisted of 15 questions about the usefulness of the JSEM-HP tool, the second block contained 15 questions about how pleasant the users think the tool is to use, and the third block contained 6 queries regarding which method of navigation was better: navigation performed in JSEM-HP (navigation restricted by knowledge) or traditional Web navigation (free navigation). Students rated each question using an ordinal scale of five items. Specifically, each question had possible answers ranging from 1 to 5 (1: "not at all", 2: "a little", 3: "some", 4: "a lot", and 5: "quite a lot").

The questionnaire responses were analyzed using the R statistical computer program (version 2.9.2). Non-parametric descriptive statistics were used to describe the main features of the results in quantitative terms. Specifically, box-and-whisker diagrams were used because the authors believe that this is a convenient way of graphically depicting groups of numerical data through their five-number summaries. The five-number summary of a data set consists of the sample minimum (smallest observation), the lower quartile or first quartile (which cuts off the lowest 25% of the data), the median (middle value), the upper quartile or third quartile (which cuts off the highest 25% of the data), and the sample maximum (largest observation).

Figure 9 shows the box plot generated for each block in the questionnaire. In each diagram, the X-axis shows the survey questions included in the corresponding block: q1 to q15 for the first block (usefulness of JSEM-HP), q16 to q30 for the second block (pleasantness of use

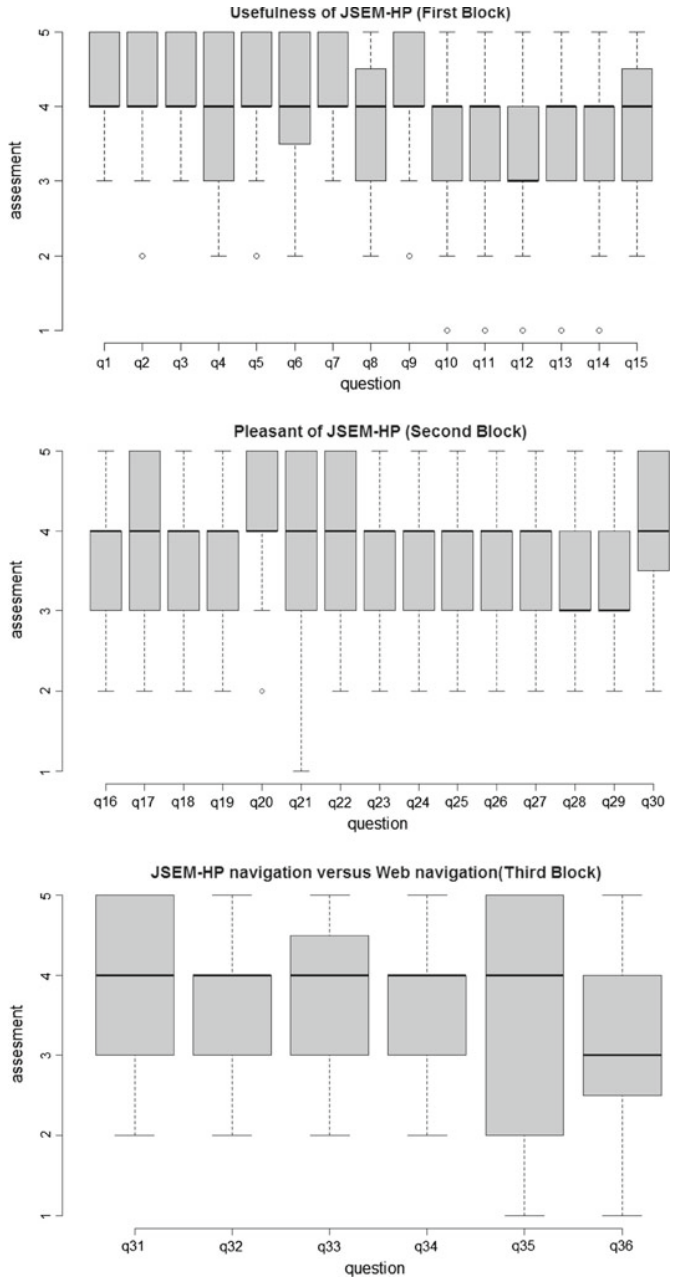


Fig. 9 Box-and-whisker diagrams

of JSEM-HP), and q31 to q36 for the third block (comparison of JSEM-HP with Web). The Y-axis shows the values of the opinion scale (1–5). The bottom and top of each box represent the lower (Q1) and upper (Q3) quartiles respectively, and the line cutting the box represents the median (Q2). As shown in the diagrams, most of the questions have Q2 = 4, which is a pretty good result.

For example, for question q4 (the question about the usefulness of hiding the informational domain in the conceptual structure used for navigation), the lower quartile is 3, the upper quartile is 5, and the median is 4. Consequently, the interquartile range is 2 (IQR= Q3 –Q1). Because the median for question q4 is right in the center of the box, the distribution is completely symmetrical for this question. In addition, the diagrams show the smallest non-outlier observation or "left whisker" (Q1 – 1.5IQR) and the largest nonoutlier observation or "right whisker" (Q3 + 1.5IQR). Finally, the diagrams also show outliers. As shown in Fig. 9, there are no "extreme outliers," only some "mild outliers." The mild outliers are values between 1.5IQR and 3IQR below Q1 and are represented by open and closed dots. An example is outlier 2 obtained for question q2 (the question about the usefulness of displaying the conceptual relationships in the navigational structure).

This information makes it possible to display the dispersion or heterogeneity of the data, which increases with the size of the box-and-whiskers. Caution must be used in interpreting this measure because the number of students surveyed was relatively small, and therefore, dispersion may be suggested where in fact there is none. For this reason, the authors conducted a further dispersion study to compute the variance (σ^2) and standard deviation (σ) for each question. This confirmed that only 8 of the 36 questions have a variance greater than one. In particular, q20 has the lowest dispersion ($\sigma^2 = 0,492$, $\sigma = 0,701$), and q35 has the highest dispersion ($\sigma^2 = 1,725$, $\sigma = 1,313$). In fact, q35 is the only question with a standard deviation greater than 1.5 in the whole sample. This result is very interesting because, as described later, question q20 was the most highly valued by the students, while q35 was one of the least highly valued.

Besides the median, the arithmetic mean ($\bar{x}$) was used as a measure of central tendency. As shown in Fig. 10, this measure was calculated for each question (36 circles) and for each block of questions (3 squares). The two continuous lines around the center line formed by connecting all the arithmetic means represent the confidence interval. The 90% confidence interval was calculated for each question from its standard deviation. The interval of each mean confirms the results of the previous analysis in terms of standard deviation and indicates the reliability of the estimate. In addition, a horizontal line has been drawn on the diagram that cuts the Y-axis at the value 3.5. It is clear that only five questions are rated on average below that value (q12, q28, q29, q35, and q36). All except q36 are above 3.4. Question q36, the worst rated in the questionnaire, obtains a score of 3.16 on average. In question q36, students were asked whether they prefer to navigate using JSEM-HP or to navigate the Web directly. This question implies an explicit and direct comparison with the navigation system that they have been using for years. In this case, to exceed the threshold value of 3 is quite an achievement.

Continuing with the discussion of arithmetic means, the aspect of the tool that was most highly evaluated was the "Existence of partial presentations of the whole conceptual domain," which obtained average scores of 4.19 for usefulness (q5) and of 4.32 for pleasantness of use (q20). The other aspects with a score higher than 4 were: "Visualization of the conceptual structure to which the available information is associated (usefulness)" (q1), "Visualization of the name of the relations among the concepts in the conceptual structure (usefulness)" (q2), "Visualization of the type of information associated with each concept in the conceptual structure (usefulness)"(q3), "Navigation through the complete conceptual structure (useful-

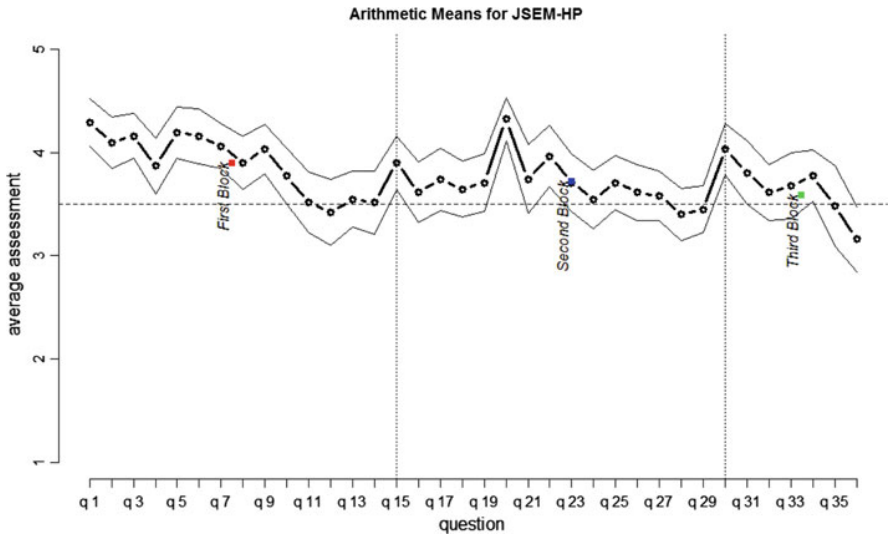


Fig. 10 Arithmetic means and confidence intervals diagram

ness)" (q6), "Navigation through the conceptual structure of a presentation (usefulness)" (q7), "Adaptation by hiding the information that is not available according to the established pedagogical prerequisites (usefulness)" (q9), and "Adaptation by special annotation of the information marked as interesting on the conceptual structure of navigation (pleasantness of use)" (q30). All aspects of the UM were given between 3.4 and 3.7 points for usefulness and pleasantness of use: "User model that stores the user's knowledge" (q10 and q25), "User model that stores the user's interests and goals" (q11 and q26), "Possibility of viewing the user model" (q12 and q27), "Possibility of modifying the knowledge inferred in the user model" (q13 and q28), and "Possibility of explicitly marking interests in the user model" (q14 and q29). The last block compared JSEM-HP's navigation and traditional navigation with regard to which is more efficient (q31), easier to use (q32), faster to use (q33), easier to learn (q34), and more attractive (q35). Among these aspects, the one that obtained the highest score was efficiency. Specifically, the students gave on average 3.8 points to question q31: "Do you consider that navigation with JSEM-HP is more efficient (it lets you find what you are looking for) than traditional Web navigation?" Overall, the block that evaluated the usefulness of the tool received a score of 3.89, the block that assessed how pleasant it is to use the tool obtained a score of 3.71, and the block that compared the tool's navigation features with traditional Web navigation obtained a score of 3.58.

The authors consider that these are good results, especially taking into account that: (a) this was the first time the students had experienced this way of navigation based on conceptual structures and the first time that JSEM-HP had been used with students, and (b) the tool, initially developed from the author's perspective, is not especially attractive from a graphical point of view.

6 Conclusions and recommendations for further work

User modeling has been performed in a number of different ways because of the wide variety of information and services offered by adaptive systems. To perceive some order in the

diversity of UMs, the authors started by presenting a general taxonomy to classify UMs from multiple perspectives (granularity, main trace, structure, context, initialization, update, stability, visibility, direction of inference, etc). This taxonomy offers a simple nomenclature and a categorization system that made it possible to position this proposal by implicitly comparing it with other related proposals according to the classification criteria established in the taxonomy.

Having established the theoretical framework, the paper presents the model, SEM-HP that defines an architecture for AHSs with two main dimensions: (a) the vertical dimension, on which the storage, presentation, navigation, and adaptation aspects of the AHS are presented as four interrelated systems, and (b) the horizontal dimension, on which a meta-level is defined above each system to guarantee its correct evolution. In addition, an incremental development process has been described that enables the definition of a wide range of navigation structures (CS_L) for which four different navigation modes are offered. Finally, the adaptation process is described on two levels: (a) adaptation to each user by means of an individual UM that takes into account personal data, experience, preferences, and interests, and (b) adaptation to a group of users by means of a group UM that captures the conceptual transitions performed by the users while they navigate in traditional mode. The individual UM makes it possible to choose the navigation structure that best suits the user, to generate personalized information summaries, and to apply orientation and advice techniques such as item annotation, item hiding and disabling, and guided routes. The group UM makes it possible to identify significant differences between the structures defined by the AHS's author and the mental structures that are common to the group of users and to suggest an appropriate evolution that would bring these two structures closer together.

To assess the model, the authors have presented a semiformal evaluation with 31 students from the University of Granada. JSEM-HP was used to create an AHS in the knowledge domain of object-oriented programming (a course taken by these students). The evaluation was successful for the 36 aspects of the JSEM-HP tool that were tested. This evaluation was carried out using an opinion questionnaire that students completed after using the system to learn the general concepts of object-oriented programming. For example, the students gave an average score of 4.29 points (maximum 5) to the usefulness of the JSEM-HP graphic browser (in the SEM-HP model, navigation is performed using the semantic net itself), 4.06 points to the usefulness of navigation on a partial view of the graph (the SEM-HP model provides the user with a different presentation (CS_P) depending on the knowledge subdomain in which the user is most interested), and 4.16 points to adaptation and user modeling. Questions were also asked to compare JSEM-HP browsing and traditional Web browsing. In this case, the students gave 3.8 points to the efficiency of finding information through the JSEM-HP browser compared to seeking information on the Web.

Immediate future work will be focused mainly in two directions: (1) For the SEM-HP model and specifically its UM, the authors will try to overcome the limitations described in Sect. 4.3, using standards that guarantee the portability of UMs managed in SEM-HP and including contextual and psychological aspects. The inclusion of user context (including psychological context) in both information and meta-information contexts [25] makes it possible to provide the most useful information to the user in both contexts. With this purpose in mind, the authors are studying several contextual information retrieval techniques and evaluation methodologies, as described in [26]; (2) With respect to the JSEM-HP tool, the authors will focus on completing its development, including several aspects that have not yet been implemented, such as guided routes and adaptation to a group of users, with a special focus on improving the user interface to increase its usability from the user/author's perspective. These efforts intend to encourage use of the tool in many other subjects at the

University of Granada, because after preliminary evaluation, it is clear that it can be used as a teaching tool. Both teachers and students that have used it regard SEM-HP as useful in the process of teaching and learning. In this context, the authors plan to reevaluate JSEM-HP by carrying out a larger quantitative study involving one group of students using the JSEM-HP tool and a second group using a traditional Web browser. The user group using the JSEM-HP tool will be split into two subgroups, one consisting of inexperienced users and the other of students who have some experience in using it.

Acknowledgments This research has been supported by the Spanish MCYT under R + D projects TIN2008-06596-C02-02, TIN2007-64718, and TIN2007-60199, by the Research Program of the Andalusia Government under project P08-TIC-03717, and by the European Regional Development Fund (ERDF).

Appendix

See Table 5.

Table 5 Questions included in the poll

	Rate on a scale of 1–5 points the **usefulness** of each of the following aspects of the JSEM-HP tool	Rate on a scale of 1–5 points the **pleasantness** of use of each of the following aspects of the JSEM-HP tool
Visualization of the conceptual structure with which the available information is associated	q1	q16
Visualization of the name of the relations among the concepts in the conceptual structure	q2	q17
Visualization of the type of information associated with each concept in the conceptual structure	q3	q18
Possibility of hiding the informational domain in the conceptual structure used to navigate	q4	q19
Existence of partial presentations of the whole conceptual domain	q5	q20
Navigation through the complete conceptual structure	q6	q21
Navigation through the conceptual structure of a presentation	q7	q22
Navigation restricted by knowledge	q8	q23
Adaptation by hiding information that is not available according to established pedagogical prerequisites	q9	q24
User model that stores the user's knowledge	q10	q25
User model that stores the user's interests and goals	q11	q26
Possibility of viewing the user model	q12	q27
Possibility of modifying the knowledge inferred in the user model	q13	q28
Possibility of explicitly marking interests in the user model	q14	q29
Special annotation of the information marked as interesting in the conceptual structure	q15	q30

Table 5 continued

	Rate on a scale of 1–5 points the following questions, **comparing** JSEM-HP navigation with traditional Web navigation (by means of links embedded within pages)
Do you think that JSEM-HP navigation is more efficient (you can find what you want) than traditional Web navigation?	q31
Do you think that JSEM-HP navigation is easier to use (find what you want without effort) than traditional Web navigation?	q32
Do you think that JSEM-HP navigation is faster to use (find what you want in less time) than traditional Web navigation?	q33
Do you think that JSEM-HP navigation is easier to learn (understanding the structure of the site) that traditional Web navigation?	q34
Do you think that JSEM-HP navigation is more attractive (more interesting or appealing) than traditional Web navigation?	q35
Do you prefer to browse the Web with the JSEM-HP tool rather than using traditional navigation?	q36

References

1. Bush V (1945) As we may think. Atl Mon 176(1):101–108
2. Nelson TH (1965) A file structure for the complex, the changing, and the indeterminate. In: Proceedings, ACM twentieth national conference, pp 84–100
3. Vogiatzis D, Tzanavari A, Retalis S, Avgeriou P, Papasalouros A (2004) The learner's mirror. In: Proceedings, ninth European conference on pattern languages of programs (EuroPLoP 2004), Kloster, Isree, Germany, July 7–11, 2004
4. Wu H, De Bra P, Aerts A, Houben G (2000) Adaptation control in adaptive hypermedia systems. Lecture notes in computer science 1892, pp 250–259
5. De Bra P, Stash N, Smits D, Romero C, Ventura S (2007) Authoring and management tools for adaptive educational hypermedia systems: the AHA! Case study. Studies in computational intelligence (SCI), No. 62. Springer, pp 285–308
6. Stash N, Cristea AI, De Bra P (2007) Adaptation languages as vehicles of explicit intelligence in adaptive hypermedia. Int J Contin Eng Educ Life Long Learn 17(4/5):319–336
7. Daoud M, Lechani L, Boughanem M (2009) Towards a graph-based user profile model for a session-based personalized search. J Knowl Inf Syst 21(3):365–368
8. Henze N, Nejdl W (1999) Bayesian modeling for adaptive hypermedia systems. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen (ABIS99), Otto-von-Guericke-Universität, Magdeburg, Germany, September 29–30
9. Heckmann D, Schwarzkopf E, Mori J, Dengler D, Kröner A (2007) The user model and context ontology GUMO revisited for future web 2.0 extensions. International workshop on contexts and ontologies: representation and reasoning (C&O:RR), Roskilde, Denmark, August 21, pp 37–46
10. Nguyen M, Santos E, Zhao Q, Hang H (2004) Capturing user intent for information retrieval. In: Proceedings, 48th annual meeting of the human factors and Ergonomics Society, New Orleans, Louisiana, September 20–24, pp 371–375
11. Brusilovsky P (2003) Adaptive navigation support in educational hypermedia: the role of student knowledge level and the case for meta-adaption. Br J Educ Technol 34(4):487–497
12. Brusilovsky P, Sosnovsky S, Yudelson M (2009) Addictive links: the motivational value of adaptive link annotation. New Rev Hypermedia Multimed 15(1):97–118

13. Zhu T, Greiner R, Haeubl G (2003) Learning a model of a web user's interests. In: Proceedings, 9th international conference on user modeling (UM'2003), Johnstown, Pennsylvania, June 22–26, pp 65–75
14. De Buenaga M, Maña M, Díaz A, Gervás P (2001) A user model based on content analysis for the intelligent personalization of a news service. In: Proceedings, 18th international conference on user modeling, Sonthofen, Germany, July 13–17. Lecture notes in computer science 2109, pp 216–218
15. Pogacnik M, Tasic J, Meza M, Kosir A (2005) Personal content recommender based on a hierarchical user model for the selection of TV programs. J User Model User-Adapt Interact 15(5):425–457
16. Komatani K, Ueno S, Kawahara T, Okuno H (2005) User modeling in spoken dialogue systems to generate flexible guidance. J User Model User-Adapt Interact 15(1):169–183
17. Ahuja A, Ng Y (2009) A dynamic attribute-based data filtering and recovery scheme for web information processing. J Knowl Inf Syst 18(3):263–291
18. Kay J (1995) The UM toolkit for reusable, long-term user models. User Model User-Adapt Interact 4(3):149–196
19. Medina-Medina N, Molina-Ortiz F, García-Cabrera L (2010) Taxonomy for user models in adaptive systems: special considerations for learning environments
20. García-Cabrera L, Rodríguez MJ, Parets J (2002) Evolving hypermedia systems: a layered software architecture. J Softw Maintenance Evol Res Pract 14(5):389–405
21. Medina-Medina N, Molina-Ortiz F, García-Cabrera L (2005) Diversity of structures and adaptive methods in an evolutionary hypermedia system. J IEEE Proc Softw 152(3):119–126
22. Medina-Medina N, Molina-Ortiz F, García-Cabrera L (2003) Personalized guided routes in an adaptive evolutionary hypermedia system. Lecture notes in computer science 2809, pp 196–207
23. Medina-Medina N, Molina-Ortiz F, García-Cabrera L (2006) An adaptation method by feedback in an evolutionary hypermedia system. In: Proceedings, 6th international conference on web engineering (ICWE 2006). Palo Alto, CA, USA, July 12–14, pp 161–168
24. Molina-Ortiz F, Medina-Medina N, García-Cabrera L (2006) An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems. In: First international workshop on adaptation and evolution in web systems engineering (AEWSE 2006), Palo Alto, CA, USA, July 12–14, No. 12
25. Motelet O, Baloian N, Pino J (2009) Taking advantage of metadata semantics: the case of learning-object-based lesson graphs. J Knowl Inf Syst 20(3):323–348
26. Tamine-Lechani L, Boughanem M, Daoud M (2009) Evaluation of contextual information retrieval effectiveness: overview of issues and research. J Knowl Inf Syst. doi:10.1007/s10115-009-0231-1

Author Biographies

Nuria Medina-Medina received the Ph.D. in Computer Science from the University of Granada in 2004. Since 2001, she works as researcher and associate professor in the Department of Computer Languages and Systems of this Spanish University. She is member of the GEDES group—research group in specification, development and evolution of software, http://www-lsi.ugr.es/~gedes. Her main research interests include hypermedia systems, user modeling, user adaptation, and software evolution. In addition, recently she has worked on other topics such as Web browsing, refactoring for visually impaired and bioinformatics.

Fernando Molina-Ortiz earned his degree in Computer Science at the University of Granada, where he worked as an associate professor from 2004 to 2007. His research interests include software evolution, user adaptation and hypermedia systems. Currently, he works as the web coordinator for the Spanish trade union CSI-F.

Lina García-Cabrera received the Ph.D. in Computer Science from the University of Granada in 2001. Since 1993, she works as researcher and Senior Lecturer in the Department of Computer Science of the University of Jaén (Spain). She was member of the GEDES group—research group in specification, development and evolution of software—and since 2009 is member of the of the SINBAD2 research group—Intelligent Systems Based on the Fuzzy Decision Analysis, http://sinbad2.ujaen.es/sinbad2/. Her main research interests include the development and application of evolutionary and adaptive hypermedia systems, application of hypermedia systems to e-learning, user adaptation, and software evolution and teaching innovation.

Diversity of structures and adaptive methods on an evolutionary hypermedia system

N. Medina-Medina, F. Molina-Ortiz and L. García-Cabrera

Abstract: SEM-HP is a model for the development of evolutionary and adaptive hypermedia systems by means of: a development process that consists in iterative and evolutionary phases; a layered architecture that captures each phase in subsystems divided into two levels of abstraction (system and metasystem); and an author tool that implements the model. The paper focuses on the capability of adaptation of the hypermedia systems developed according to the SEM-HP model, in particular, the learning subsystem that is in charge of performing the user adaptation. SEM-HP offers two important contributions: supporting the process of evolution of the development of adaptive hypermedia systems, ensuring easy, flexible and consistent maintenance; and the orientation support is that inherent to the navigational structure itself.

1 Introduction: the SEM-HP model

SEM-HP [1] is a model for the development of hypermedia systems (HS) with antecedents in the Wang [2] and Stotts [3] approaches. SEM-HP provides to the author the necessary formalisms, techniques and methods to support two essential aspects: evolution and adaptation. The *ability to evolve* eases the creation of the HS and guarantees its long-term life. The evolutionary mechanisms provided by SEM-HP allow the HS to incorporate the needed changes in an easy, flexible and consistent way. The *ability to adapt* the HS to the needs and features of each user is essential if we take into account that there will be users with very different profiles accessing our system. This process can be seen as a particular case of evolution where the system automatically changes its behaviour depending on the user utilising it [4].

SEM-HP model provides the author with three elements for the creation of evolutionary and adaptive hypermedia systems: a development process, an architecture and an author tool. The development process establishes the guidelines to follow for the creation of the HS from a software engineering approach. The architecture describes the representation models used to capture each phase of the development process. Finally, the author tool eases the creation of the system according to the architecture and the development process proposed in the model.

In this paper we firstly outline the SEM-HP architecture, describing with more detail the learning subsystem, which takes care of the user adaptation and maintains a user model, and update and knowledge rules. After that, we describe the adaptive methods and techniques used in SEM-HP (choosing the appropriate subdomain and sets of rules, and personalisation to user knowledge, his interests and goals). Finally, conclusions, related work and further work are discussed.

IEE Proceedings online no. 20050026

doi: 10.1049/ip-sen:20050026

Paper received 8th April 2005

N. Medina-Medina and F. Molina-Ortiz are with the Depto. L.S.I., E.T.S.I. Informática, Universidad de Granada, Spain

L. García-Cabrera is with the Depto. Informática, Universidad de Jaén, Spain

E-mail: nmedina@ugr.es

2 SEM-HP architecture

The architecture proposed by the SEM-HP model (Fig. 1) is structured in layers, performing a double division [5]: vertical and horizontal. The vertical division considers four subsystems: memorisation subsystem, presentation subsystem, navigation subsystem and learning subsystem. The horizontal division distinguishes two layers in each subsystem: system and metasystem. The least abstract level (system) comprises the representation models defined by the author during the corresponding development phase, while the most abstract level (metasystem) includes the evolutionary mechanisms that will permit the integration and propagation of changes performed by the author in the elements of the corresponding subsystem.

The evolutionary mechanisms included in the metasystem are mainly three [6]: evolutionary actions, restrictions and change propagation. The author interacts with the metasystem to build and modify the HS. To perform a change in a subsystem the author must choose the appropriate evolutionary action and run it. To ensure the integrity of the HS, an evolutionary action is only executed if it satisfies a set of restrictions imposed by the model and by the author. In addition, when changing an element in a subsystem, the need may arise to modify other elements in the same subsystem or even in other subsystems. This change propagation is automatically done in the model, guaranteeing, in addition to the consistent evolution in each subsystem, the coevolution of the set of subsystems.

In this paper we focus on the capability of adaptation of the HS developed according to the SEM-HP model, so we will briefly outline the three first subsystems [5], and we will describe in depth the learning subsystem.

The *memorisation subsystem* stores, structures and maintains the conceptual and informational domain of the HS. The representation model used for describing both domains is a conceptual structure, CS_M (Fig. 2), or semantic net with two kinds of nodes: concepts (C) and items (I). Items are informational units (text, image, audio, video, executable, etc.). The concepts are connected between them

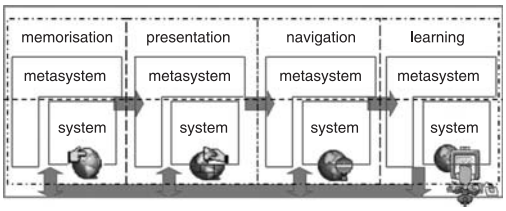


Fig. 1 *SEM-HP architecture for a hypermedia system*

through semantically labelled relations, called conceptual relations. The informational items are associated with concepts; in this way the links between items are established based on the semantic relations among concepts. The association between an item and a concept is called functional association and it is labelled with the role the item plays with respect to the concept. In addition to the role, each item has some properties that determine the main features of the information it contains: author, date of creation, level of difficulty, media (audio, text, video, etc.), language, etc.

In the *presentation subsystem* the author selects different subsets of the CS_M created in the memorisation phase, with the aim of reducing its complexity and size (Fig. 3). Creating different views of the CS_M achieves: (*a*) provision to the user of a navigation structure focused on the knowledge subdomain in which he is interested; and (*b*) reduction of the disorientation problems. Based on the same knowledge domain (conceptual and informational domain captured in a CS_M), the author prepares different conceptual structures of presentation (CS_P). The author is in charge of defining the subdomains and of labelling each CS_P with the subdomain it represents.

In the *navigation subsystem* the author establishes the navigability of the conceptual relations. By default, the conceptual relations will be navigated from the origin to the destination concept. Nevertheless, if the author wishes, he can extend the navigability of a conceptual relation in the other direction, so that it can be navigated in both ways. The author defines the navigational possibilities for each presentation, being able to create different conceptual structures of navigation (CS_N) from the same CS_P.

3 Learning subsystem

The learning subsystem permits the HS developed according to the SEM-HP model to be qualified as *adaptive*. In the two subsystems described above the author prepares different navigation structures for the same knowledge domain. These structures differ in the knowledge subdomain they represent and in the way their conceptual relationships can be navigated. Because of this diversity, the system can offer to the user the navigation structure that best matches his profile this being understood as several general features, such as the subdomain he is interested in, experience in the subject and experience in navigation. Two users with the same profile will obtain the same structure – nevertheless they are different. For example, although both may be interested in the same subdomain, their knowledge goals can be different, or, even if both have similar average experience in the subject, one may know some concepts better than the other. Hence, individual adaptation methods and techniques are needed, which in the SEM-HP architecture are included in the learning subsystem. The elements the learning subsystem uses to apply these adaptation methods are: a user model and a set of learning rules (update rules and knowledge rules). Again, for the same CS_N the author can define different sets of learning rules, creating different conceptual structures of learning (CS_L).

3.1 User model

The user model is the internal representation of the user that the learning subsystem keeps. The information the user model contains is shown in Table 1. Most of the initial information is explicitly set by the user the first time he logs on to the system. Part of that information will stay unchanged until the user requests to change it, while other information will be automatically updated as the user browses the HS. So, we can characterise the HS developed according to the SEM-HP model as adaptable and adaptive (following the definition of both terms shown in [7]).

The user will choose the knowledge subdomain in which he is most interested, selecting one of the subdomains defined by the author in the presentation phase. Whenever he wishes he will be able to choose another subdomain by a direct petition to the system. The initialisation and update tasks are performed automatically for other entries in the user model, such as the knowledge degree and the number of visits. The first time the user logs on to the system the number of visits is 0 and his knowledge about every item in the HS is set to 'null'. While the user navigates, the system counts his visits and updates the degree of knowledge about the visited items and other items related to them using the update rules. There are other entries which are updated by the system upon the user's request. This is the case of the

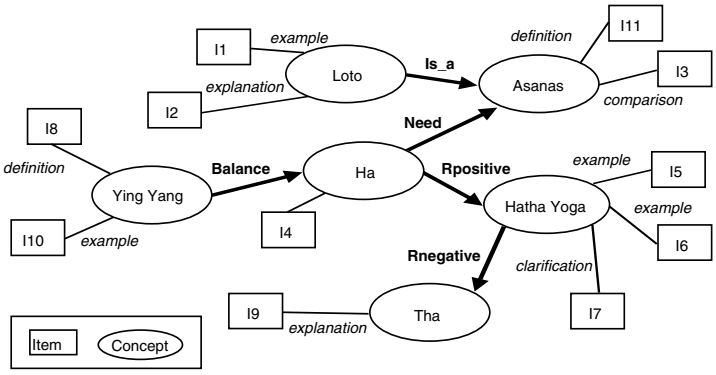


Fig. 2 *Conceptual structure of memorisation*

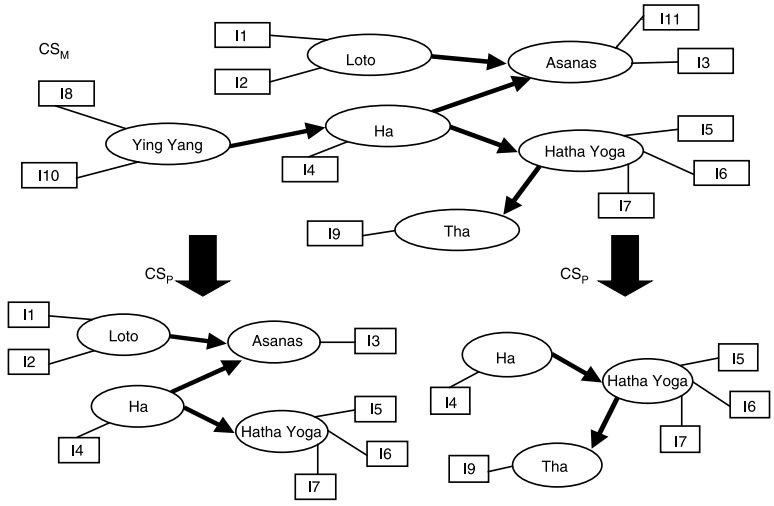


Fig. 3 *Two different views of the same CS_M*

Table 1: User model

		Initialisation	Update
Subdomain of interest		Asked of the user	Performed by the user
Experience in the subject		Asked of the user	Requested by the user
Experience in navigation		Asked of the user	Requested by the user
For each item	Degree of knowledge	Automatic	Automatic
	No. of visits	Automatic	Automatic
Interests / goal		Asked of the user	Performed by the user
Preferences		Asked of the user	Performed by the user
Personal information		Asked of the user	Performed by the user

experience of the user in the subject and in navigation. Initially the system can only obtain this information by explicitly asking the user, but later the system can automatically infer these data from other information in the user model. The subject experience can be approximated as the average user knowledge, and the navigation experience can be inferred from the total number of visits performed.

3.2 Update rules

The degree of user knowledge about an item I, $K(I)$, is represented in the user model using any of the semantic labels ($label_{SEM}$): 'null', 'very low', 'low', 'medium', 'high', 'very high' and 'total', each one with an associated numerical value from one to seven. The update rules (R_u) determine how, after every visit the user carries out, his degree of knowledge about certain items is increased. There will be an R_u associated with the visit to each item in the CS_L. In it, the head is the visited item and the body includes a set of update predicates that will be executed after the visit. Equation (1) shows the generic structure of an R_u:

$$\text{Visit}(I) \rightarrow \text{Update}(I),\ \text{Update}(I'') \ldots \text{Update}(I''') \quad (1)$$

Each one of the predicates updates a different item in the CS_L. The update of the user knowledge about an item can be: incremental or fixed (the degree of user knowledge is set to a fixed degree of knowledge), absolute (it uses an absolute $label_{SEM}$) or relative (it uses the degree of user knowledge about the visited item), each-time or first-time (it is run only the first time the item in the head is visited). Table 2 shows the available update predicates and Table 3 includes examples of updates to the item *I*6 after *I*4 is visited.

To help the author, the system generates by default a set of R_u for each CS_L. In the body of an R_u there is only one

Table 2: Update predicates

		Each-time	First-time
Incremental	Absolute	Ink-abs (*I*, number-of-degrees)	Ink-abs_{first} (*I*, number-of-grades)
	Relative	Ink-rel (*I*, number-of-degrees)	Ink-rel_{first} (*I*, number-of-degrees)
Fixed	Absolute	Fix-abs (*I*, $label_{SEM}$)	Fix-abs_{first} (*I*, $label_{SEM}$)
	Relative	Fix-rel (*I*)	Fix-rel_{first} (*I*)

Table 3: Some possible updates on the item I6 after the visit to I4

	Update(I6)	K(I6) after visit to I4
Current User Knowledge: $K(I6) =$ 'null', $K(I4) =$ 'medium'	Ink-abs(I6, 2)	$K(I6) = K(I6) + 2 =$ 'low'
	Ink-rel(I6,2)	$K(I6) = K(I4) + 2 =$ 'very high'
Update Rule: Visit $(I4) \rightarrow \ldots,$ Update$(I6), \ldots$	Fix-abs(I6, 'total')	$K(I6) =$ 'total'
	Fix-rel(I6)	$K(I6) = K(I4) =$ 'medium'

update predicate by default, which sets to 'total' the degree of knowledge about the visited item. For example: Ru$(I4)$: *Visit*$(I4) \rightarrow$ *Fix-abs*$(I4,$ *'total'*$)$. The author can make the default update rules evolve, adding updates about other items different from the visited item. All the changes are done through evolutionary actions, so they will not be run unless they satisfy several restrictions, such as not removing the visited item's update or not including two updates about the same item. The restrictions guarantee that the set of R_u satisfies some desirable features, such as that every R_u must permit the user to achieve a 'total' knowledge about the head item after a finite number of visits to it. For example, by performing valid changes, the author can transform the default rule for *I*4 into this other (the changes are in boldface): Ru$(I4)$: *Visit*$(I4) \rightarrow$ Fix-abs$(I4,$ 'total'$)$, **Ink-abs(*I*6, 2), Ink-abs(*I*7, 3)**.

3.3 Knowledge rules

The author defines, for each CS_L, a set of knowledge rules (R_k). These R_k will be used to adapt the user navigation depending on his degree of knowledge. Hence, the R_k associated with an item determines which other items must be known by the user and which degree of knowledge he must have about them in order to access the item. Equation (2) shows the structure of an R_k and Table 4 contains the allowed knowledge restrictions:

$$\begin{aligned} &\text{Knowledge\_restriction}(\mathrm{I}')\ \mathrm{op}_\mathrm{L} \ldots \mathrm{op}_\mathrm{L} \\ &\text{Knowledge\_restriction}(\mathrm{I}'') \rightarrow \text{Visitable}(\mathrm{I}) \\ &\text{where } \mathrm{op}_\mathrm{L} \in \{\text{and, or}\} \end{aligned} \tag{2}$$

Once more, to assist the author, the system generates an set of default R_k, which can be changed by the author using the appropriate evolutionary actions. For each item in the CS_L the system generates an R_k for each conceptual relation which can be navigated up to the concept to which the item is associated. The body of the rule expresses the requirement of a 'total' knowledge about any of the items associated with the origin concept of the relation. Figure 4 shows the R_k automatically generated in the example.

Some modifications the author can carry out in an R_k are: remove an R_k, change the logical operators that connect the knowledge restrictions, add new restrictions or change the restriction about an item. For example, the author can change the default R_k for I_3 (Fig. 4), producing the R_k shown in Table 5.

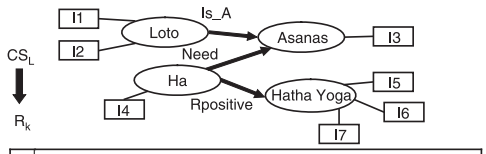
Before performing the changes requested by the author, the metasystem checks that the integrity restrictions hold. For example, the set of R_k must guarantee that performing the appropriate navigation steps, every item in the CS_L will become accessible (when the navigation is adapted to the user knowledge, an item is accessible if the user fulfils at least one of the R_k associated with it). This restriction will depend not only on the R_k but also on the R_u. To check it, the system builds a knowledge navigation tree that permits one to check that all the items are accessible, so as a result it

Table 4: Knowledge restrictions in the body of R_k

	Strict	Not strict
Equality	$K(I) = \text{label}_{SEM}$	
Larger	$K(I) > \text{label}_{SEM}$	$K(I) >= \text{label}_{SEM}$
Smaller	$K(I) < \text{label}_{SEM}$	$K(I) <= \text{label}_{SEM}$
Interval	$I1 < K(I) < I2$	$I1 <= K(I) <= I2$
	where I1 and I2 are label_{SEM} and $I1 < I2$	
OR_{SET}	$K(I) \propto$ [restriction$_1$, restriction$_2, \ldots,$ restriction$_N$] which means $K(I)$ satisfies at least one restriction in the set	

Table 5: Changes in the R_k associated with the visit of I3

$Rk^1(I3)$: $K(I1) >$ 'null' or $K(I2) >=$ 'medium' → Visitable $(I3)$ [IsA]
$Rk^2(I3)$: $(K(I2) >$ 'high' or $K(I6) >$ 'high') and $K(I4) >$ 'high' → Visitable $(I3)$ [Need]



I1	Without restrictions
I2	Without restrictions
I3	Rk^1(I3): K(I1) = 'total'or K(I2) = 'total' → Visitable (I3) [Is_A] Rk^2(I3): K(I4) = 'total' → Visitable(I3) [Need]
I4	Without restrictions
I5	Rk^1(I5): K(I4) = 'total' → Visitable(I5)
I6	Rk^1(I6): K(I4) = 'total' → Visitable(I6)
I7	Rk^1(I7): K(I4) = 'total' → Visitable(I7)

Fig. 4 *Default knowledge rules*

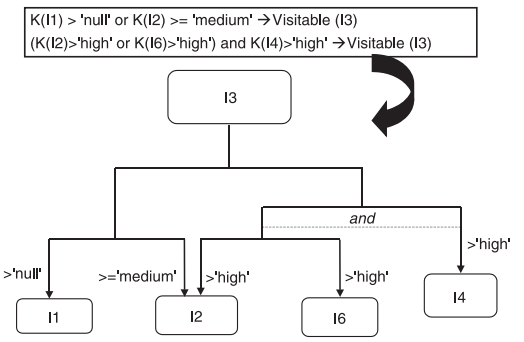


Fig. 5 *Tree$_k$ for the item I3*

is possible to reach a 'total' knowledge about each one of them (more details in Section 4.2).

Every R_k can be expressed as a knowledge restriction tree ($Tree_K$). The head item of the R_k is the root of the $Tree_K$. The composite restrictions (restrictions joined by logic operators) in the body of the R_k are decomposed in the restrictions that integrate it, generating subtrees inside the $Tree_K$. Every single knowledge restriction is represented by a child node labelled with the item implied in the restriction. The branch getting to the node is labelled with the knowledge degree required for the item in the restriction. To represent the *and* operator we join the implied branches with a horizontal dotted line. All the R_k associated with an item can be unified in a single tree using *or* branches, since it is enough that one R_k holds for the item to be accessible. Figure 5 shows the $Tree_K$ that will allow one to determine whether *I*3 is accessible.

4 Adaptive methods and techniques

4.1 Choosing the CS_L

In each development phase the author prepares different CSs, starting from each of the CSs defined in the previous phase. For example, if the author builds two CSs based upon each CS in the preceding phase, the number of CS_Ls available to be browsed by the user will be 2^3 (Fig. 6).

During the development process of the HS the author must label the created CSs so that the system can automatically determine which is the one that best fits the user. There are three labels for each CS_L:

- *Knowledge subdomain*: The knowledge subdomain shown by a CS_L is established in the presentation phase, since in this phase the author perfectly knows which part of the knowledge domain represented in the CS_M he has captured in the presentation.
- *Experience in navigation*: The value of this label is defined in the presentation and navigation phases. For example, a conceptual structure with a considerable amount of items, concepts and extended conceptual relations offers very many navigation possibilities, and consequently users with little experience in navigation are more likely to feel lost.
- *Experience in the subject*: The value of this label is determined by design decisions taken by the author in the presentation and learning phases. For example, with restrictive knowledge rules and update rules that quickly increase the user's knowledge, a high level of experience is required.

Since there are three attributes in the user model with the same name, meaning and value range, the chosen CS_L for each user is the one whose values in these three labels are closer to the values of the corresponding attributes in his user model.

4.2 CS_L personalised to user's knowledge

During the user navigation, the system personalises the CS_L through the annotation of the visited items and the visualisation in the CS_L itself of the user's knowledge state. The annotation of the previously visited items with a special colour permits us to avoid undesired repeated visits. The visual labelling of the items with the knowledge degree the user has about them allows him to check how his knowledge increases as he browses the HS. This makes the user aware of the knowledge prerequisites of the system and of his learning process.

In addition, the system uses the item hiding technique, in which forbidden items are hidden and disabled, ensuring that the user only visits accessible items according to the R_k defined in CS_L. With this adaptive technique the visible items in the CS_L depend on such an individual thing as the knowledge state of the user, restricting the navigation in the function of his knowledge. In Fig. 7 the adaptation of a CS_L after visiting the item *I*4 is shown.

To calculate the accessible items, the R_k must be evaluated on the user's current knowledge state. To efficiently perform this task, the system creates a knowledge navigation tree ($Navigation_K$ Tree) for each CS_L, applying its R_k and R_u. Each node in the tree represents a knowledge state in which the user can be, that is, the degree of knowledge the user has in a particular navigational step about every item in the CS_L. The root node represents total ignorance ('null' knowledge about all the items), and the leaf nodes represent 'total' knowledge about all the items. Each node has as many children as items are accessible from the knowledge state represented by it. The branch that leads to each child node is labelled with the name of the visited item and the child node is built applying the visited item's R_u on the knowledge state represented by the parent node. In Fig. 8 we can see part of the $Navigation_K$ Tree generated for the CS_L in Fig. 7, using the default R_u and R_k, after applying to them the changes shown in Sections 3.2 and 3.3 for $R_u(I4)$ and $R_k(I3)$. The knowledge state of a user who has only visited *I*4 is marked in the Figure.

Once the $Navigation_K$ Tree has been generated for a CS_L, it is used to restrict the navigation by knowledge of all the users who browse it. This tree will only be changed when the R_u or the R_k that were used to generate it change, be it requested by the author through an evolutionary action or caused by an automatic change propagation carried out

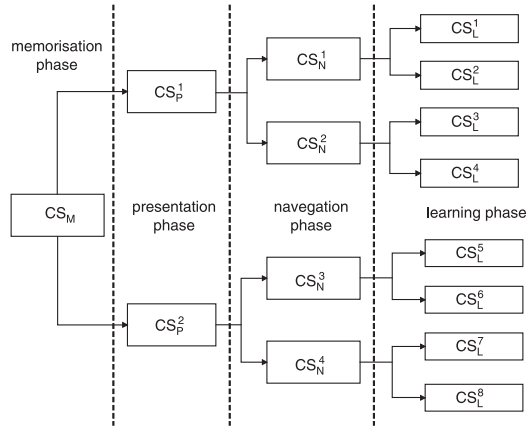


Fig. 6 *Eight different CS_L*

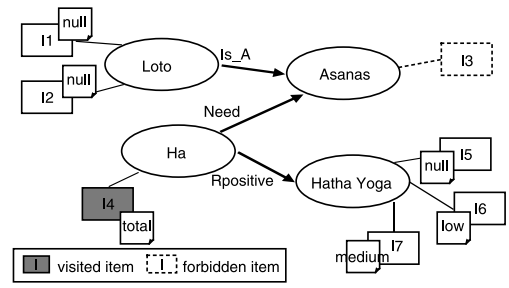


Fig. 7 *CS_L adapted to user's knowledge*

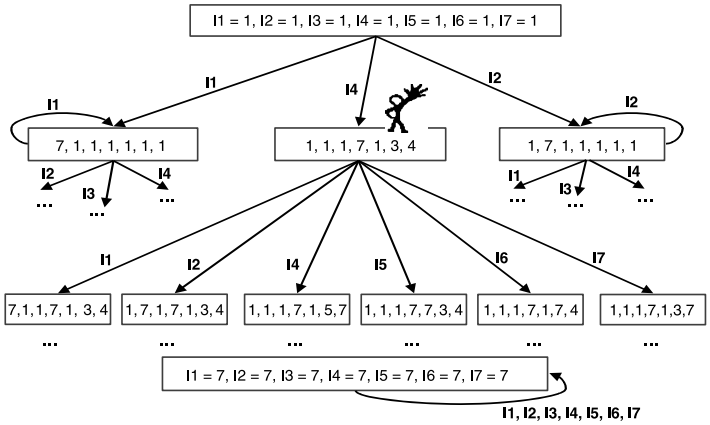


Fig. 8 *Navigation$_k$ tree*

by the metasystem. Every time the tree is generated, the system checks restrictions to guarantee that the set of R_u and R_k is consistent. For example, the presence of leaf nodes representing 'total' knowledge ensures that this knowledge state is reachable by the user.

4.3 CS_L personalised to user interests

During the choice of the CS_L, the system takes into account the knowledge subdomain in which the user is interested, contemplating to some extent his interests and wishes. Nevertheless, inside a knowledge subdomain there can be many items, and the user will probably be more interested in a group of them.

Because of this, the user can specify his interests as the set of items that he is most interested in learning (interesting items). Hence, during navigation, the system will annotate in a special way the items of the CS_L that will help the user to achieve his interests (desirable items). An item is desirable by the user if it satisfies two properties: (i) it is accessible; and (ii) it is interesting or its visit will directly or indirectly contribute to turn an interesting item into accessible. Therefore, when visiting a desirable item the user knows that he is visiting an item he defined as interesting, or visiting an item that will bring him closer to the knowledge state required to be able to visit one or several of his interesting items.

If all the interesting items are accessible, the set of desirable items matches the set of interesting items. If not, to calculate the set of desirable items, the system builds a tree with the knowledge restrictions that are not satisfied by the user and are nevertheless necessary for him to be able to visit the interesting items. This tree is called $Tree_D$ and its leaf nodes represent desirable items. The $Tree_D$ is automatically generated using the R_k represented as $Tree_K$. In the first level of the $Tree_D$, the interesting items specified by the user are joined by an *and* connector. Until all the leaf items are marked as desirable, we proceed as follows: If a leaf item is accessible it is marked as desirable; if not, it is replaced by its $Tree_K$, removing the branches representing knowledge requirements that the user's current knowledge state satisfies.

In Fig. 9*a* we show the $Tree_D$ that is generated for the interests set Ints $= \{I3, I7\}$ being the current user knowledge marked in Fig. 8. We can see how the item $I3$, which is not currently accessible, is expanded using its $Tree_K$ (Fig. 5). Only the requisites not fulfilled by the user are included, so the branch corresponding to the $K(I4) >$ 'high' requisite is not added. Figure 9*b* shows the CS_L adapted to the user interests by annotating the desirable items (the name of the item is underlined and in a larger font).

Obviously, until all the interesting items are marked as desirable, the $Tree_D$ is updated each time the user visits an item and its R_u is run. The knowledge restrictions fulfilled by the new knowledge state are removed, and if a node becomes accessible, it is marked as desirable and the subtree under it is removed.

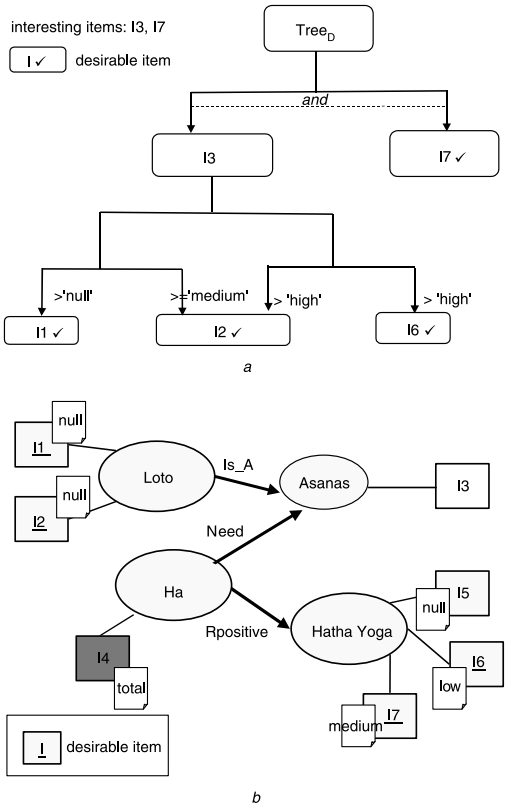


Fig. 9 *Tree$_D$ for Ints $= \{I3, I7\}$ and annotation of desirable items*

The user can also specify a knowledge goal, indicating not only the items he wishes to know but also the degree of knowledge he wants to achieve in each of them. The user will specify his goal as a set of subgoals in the form (I, label_{SEM}), where I is an item in the CS_L he browses and label_{SEM} represents the desired knowledge degree. To personalise the CS_L in the function of the user goal, the procedure explained previously is used. The only difference is that, in this case, the branches that get to the nodes placed in the first level on the Tree_D (the items included in the goal) are labelled with the knowledge restriction specified in the corresponding subgoal. Now, the knowledge restrictions in the first level are also evaluated. In this way, every time a subgoal is satisfied it is removed from the tree. When the root node of the $Tree_D$ is removed it will mean that the user goal has been completely achieved.

In addition, the user can request from the system a navigation route that brings him to his goal. This route is shown as a list of items and an order to visit them. To calculate it the system uses the Navigation_K Tree (Section 4.2), taking into account the knowledge state of the user and his preferences on the items' idiom, length of the route, etc. The details of this technique are given in [8].

5 Conclusions and related work

SEM-HP is a systemic, semantic and evolutionary model that permits the development of adaptive and evolutionary HS. The architecture of the developed systems is divided into four subsystems, with two abstraction levels in each subsystem: the higher level (metasystem) provides the necessary mechanisms to evolve the structure and the functioning of the representation models included in the lower level (system). Among the four subsystems, the learning subsystem is in charge of performing the user adaptation and to do this, it defines, among other mechanisms, a set of rules that calculate the user knowledge (R_u) and use it to restrict the user's navigation (R_k). Three types of adaptive methods are applied during the user navigation: orientation support, personalised views and guidance. Table 6 shows the adaptive techniques used to implement each one of them.

Since the first adaptive hypermedia systems (AHSs) appeared in the late 1980s, many authors have contemplated navigation support methods in their systems, models and architectures. The particularity of the SEM-HP model lies mainly in that it offers to the author support during the process of evolution of the developed AHS, ensuring an easy, flexible and consistent maintenance task.

In addition, SEM-HP integrates an important number of navigation adaptation methods described by Brusilovsky in [9], while most of the AHSs we have found focus only on some of them. The way in which SEM-HP implements each adaptive method is also different. For example, while most adaptive architectures place on the navigational structures a method that provides orientation support (for example, abstracts views in AHAM [10]), in SEM-HP the orientation support is inherent to the navigational structure itself, a separate method not being necessary.

Another differentiating aspect of SEM-HP is that the navigation structure displays, in addition to the items of information, the concepts they describe and the relations between them. Although it is true that other systems, such as the one proposed in [11], also incorporate concepts, the difference lies in the homogeneous way SEM-HP treats them. For example, the system is able to determine the knowledge the user has about the concepts using a set of weight rules (R_w) [12] previously defined by the author.

Table 6: Adaptive techniques and methods

Adaptive method		Adaptive technique
Orientation support		Annotation of visited items
		Conceptual structure showing the position of the last visited item and its context in the information/conceptual domain
Personalised views		Choice of the navigation structure according to the subdomain of interest, navigation experience and subject experience
		Annotation of degree of user knowledge about the items
		Hiding the forbidden items
		Annotation of interesting items
		Annotation of goal items
Guidance	Global	Guidance for a knowledge state taking into account user preferences
	Local	Annotation of interesting items and goal items
To the user group		Transition matrices

Each R_w calculates the user knowledge about a concept considering the user knowledge about the items associated with it. It permits using knowledge about concepts in the definition of R_k, user interests, etc.

Regarding the adaptation to the group of users, we follow an approach similar to the one used by Bollen and Heylighen [13], although with some differences. While Bollen and Heylighen apply a set of learning rules (frequency, transitivity and symmetry) to automatically strengthen or to create new hyperlinks, we build transition matrices [14], which reflect the number of times the user goes from one concept to another in each navigation

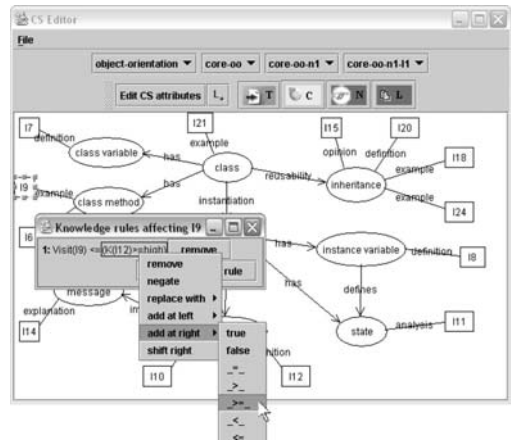


Fig. 10 *The author is writing $Rk^1(I9)$*

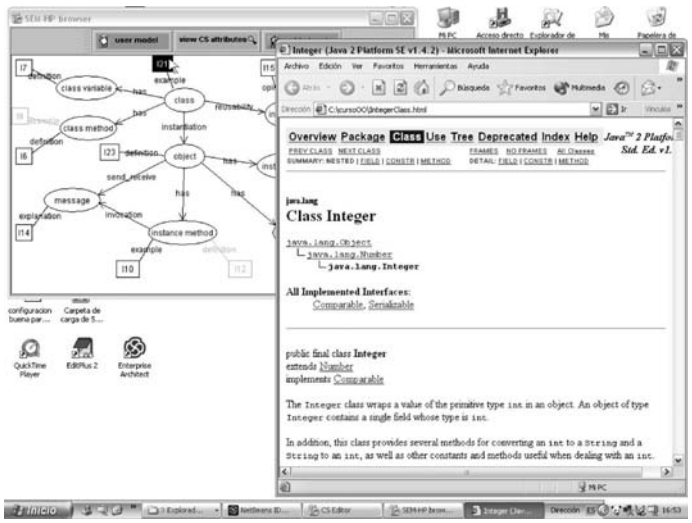


Fig. 11 *The user has selected I21*

structure [15]. Based on a matrix, the author will decide if the corresponding structure will be totally or partially adapted to the navigation pattern defined by its users. For example, he can remove a conceptual relation never used or add a relation frequently followed.

6 Further work: prototype JSEM-HP

Our further work is focused on finishing the development of a prototype that implements SEM-HP. The prototype in development is called JSEM-HP, and it is written in Java. Firstly, we are developing the author tool that will allow the developer to build and evolve adaptive hypermedia systems. JSEM-HP explicitly supports the layered division in systems and metasystems, and for now it implements the memorisation, presentation and navigation subsystems and part of the learning subsystem. For the visual manipulation of graphs it uses the Jgraph library [16]. Figure 10 shows the prototype interface when the author is editing the first knowledge rule associated with the item *I*9 in a conceptual structure whose knowledge domain is the object orientation paradigm.

Figure 11 shows the prototype interface during the user navigation – the moment in which the user has selected the accessible item *I*21. The degree of knowledge about the items is reflected through the colour of the item (darker means higher knowledge).

More details about the existing versions of the prototype are described in [14, 17]. Our aim is to use the prototype to validate the SEM-HP model, mainly in two aspects:

- *Development of AHS*: We are checking that our evolutionary approach actually helps the author to build AHS in a flexible and consistent way, i.e. the four subsystems keep consistent during their evolution, and the evolutionary actions and the restrictions are the appropriate and flexible enough to allow the author to develop and evolve AHS easily.
- *User adaptation*: Our objective is to check the different adaptation techniques implemented and verify that they enhance the navigation for heterogeneous users, improving the comprehension of the offered material and reducing the disorientation problems.

7 References

1 García, L., and Parets, J.: 'A cognitive model for adaptive hypermedia systems'. 1st Int. Conf. on WISE, Workshop on World Wide Web Semantics, Hong-Kong, China, June 2000, pp. 29–33
2 Wang, W., and Rada, R.: 'Structured hypertext with domain semantics', *ACM Trans. Inf. Syst.*, 1998, **16**, (4), pp. 372–412
3 Stotts, P., Furuta, R., and Ruiz, C.: 'Hyperdocuments as automata: verification of trace-based browsing properties by model checking', *ACM Trans. Inf. Syst.*, 1998, **16**, (1), pp. 1–30
4 Medina, N., García, L., Torres, J., and Parets, J.: 'Evolution in adaptive hipermedia systems'. Principles of Software Evolution IWPSE. Orlando, Florida, USA. May 2002, pp: 34–38.
5 García, L., Rodríguez, M.J., and Parets, J.: 'Evolving hypermedia systems: a layered software architecture', *J. Softw. Maint. Evol. Res. Pract.*, 2002, **14**, (5), pp. 389–405
6 García, L., Rodríguez, M.J., and Parets, J.: 'Formal foundations for the evolution of hypermedia systems'. 5th Eur. Conf. on Software Maintenance and Reengineering, Workshop on FFSE, IEEE Press, Lisbon, Portugal, March 2001, pp. 5–12
7 Wadge, W.W., and Schraefel, M.C.: 'A complementary approach for adaptive and adaptable hypermedia: intensional hypertext'. 12th ACM Conf. on Hypertext and Hypermedia, 3rd Workshop on Adaptive Hypertext and Hypermedia, Aarhus, Denmark, August 2001, pp. 327–334
8 Medina, N., Molina, F., and García, L.: 'Personalized guided routes in an adaptive evolutionary hypermedia system', *Lect. Notes Comput. Sci.*, 2003, **2809**, pp. 196–207
9 Brusilovsky, P.: 'Methods and techniques of adaptive hypermedia', *User Model. User-Adapt. Interact.*, 1996, **6**, pp. 87–129
10 Wu, H., and De Bra, P.: 'Link-independent navigation support in web-based adaptive hypermedia'. Web-Engineering Track of the 11th Int. WWW Conf, Honolulu, Hi, USA, May 2002, pp. 74–89
11 Da Silva, P., Durm, R., Duval, E., and Olivié, H.: 'Concepts and documents for adaptive educational hypermedia: a model and a prototype'. 2nd Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, USA, June 1998, pp. 35–43
12 Medina, N., García, L., Rodríguez, M.J., and Parets, J.: 'Adaptation in an evolutionary hyperpedia system: using semantic and Petri nets', *Lect. Notes Comput. Sci.*, 2002, **2347**, pp. 284–295
13 Bollen, J., and Heylighen, F.: 'A system to restructure hypertext networks into valid user models', *New Rev. HyperMed. Multimed.*, 1998, **4**, pp. 189–213
14 Medina, N.: 'An integral and evolutionary model of adaptation for hypermedia systems. The learning system of SEM-HP' (in Spanish). Doctoral thesis, University of Granada, Spain, November 2004
15 Salmerón, L., Cañas, J., Gea, M., Fajardo, I., Antolí A., and Abascal J.: 'Analysis of the knowledge adquisition in hypertext systems from the user's navigation strategies'. (in Spanish). COLINE, Granada, Spain, November 2002
16 Jgraph Swing Component. http://www.jgraph.com
17 Molina, F., García, L., Medina, N., and Hurtado, MV.: 'Evolutionary conceptual structure editor: practical considerations'. (in Spanish). 3rd Jornadas de trabajo DOLMEN JISBD, El Escorial, Madrid, Spain, November 2002, pp: 77–82

Aplying a Semantic Hypermedia Model to Adaptive Concept Maps in Education

Fernando Molina-Ortiz[1], Nuria Medina-Medina[2] and Lina García-Cabrera[3]

[1,2]University of Granada. E.T.S.I.I.T. 18071, Granada, Spain
[3]University of Jaén. Campus Las Lagunillas. 23071, Jaén, Spain
{[1]fmo,[2]nmedina}@ugr.es. [3]lina@ujaen.es

Abstract. We present a model and a tool that allow the development of concept maps that can be navigated by the student, thereby allowing plenty of course contents to be browsed through a concept map. The tool applies several user adaptation techniques that reduce disorientation and cognitive overhead when browsing large maps by adapting the concept maps to the student's knowledge. In addition, an evolutionary framework is used so consistency after changes is always guaranteed. The paper also describes a preliminary application to a real case at a University course.

1 Introduction

Concept Maps are diagrams that represent organized knowledge, showing concepts and the relations between them [1]. They have been shown to be very useful in education [2], and we believe that the utilization of Information Technology and Computer Science to support the use of concept maps further enhances the educational experience.

For us, a navigable concept map is a concept map that is able to present additional information when the user selects one of its elements. In this way, if the user is interested in a concept, the system allows to visualize one or more electronic documents or web pages that permits the user to learn about the concept. In an educational context, this allows to present to the user (the student in this case) plenty of course contents, which is given structure and coherence through a concept map. Therefore, the student can explore (navigate) the educational contents in a non-linear way, choosing first the elements associated to the concepts in which he is most interested, and following the strategy he prefers.

Nevertheless, this freedom in exploring the course contents can lead to problems. The student can feel overwhelmed by a big or unknown concept map, which may show too many elements at once or provide material that the student is not yet prepared to understand, leading to a helpless and demotivated student. In order to reduce these problems we apply techniques from the field of adaptive hypermedia systems [3], so we can provide concept maps that are adjusted to the student's features and knowledge, offering only the information that the student is ready to comprehend.

In this paper we present a model, SEM-HP [4,5], and an author and browsing tool, JSEM-HP [6]. They have their origin in the field of hypermedia systems,

and provide a formal model and an environment for the development of adaptive hypermedia systems based on navigable concept maps. Section 2 introduces SEM-HP and JSEM-HP, and section 3 describes a preliminary case example. Finally, related work, conclusions and further work are discussed in section 4.

2 The Model SEM-HP and the Tool JSEM-HP

SEM-HP [4,5] is a Systemic, Semantic and Evolutionary model that can be used for the development of educational systems, in which concept maps are key elements and that supports user adaptation. JSEM-HP [6] is a tool that implements the main elements of this model.

A SEM-HP system is composed by for interacting subsystems: memorization subsystem, presentation subsystem, navigation subsystem, and learning subsystem. The memorization subsystem stores, structures, and maintains the contents that will be offered to the students in the form of a global navigable conceptual map. The presentation subsystem allows the author to select subsets of a conceptual map in order to create more focused and simpler conceptual maps. The navigation subsystem provides tools to decide the order in which the information can be browsed, and the learning subsystem takes care of the user modeling and user adaptation.

SEM-HP also supports the evolution of the systems created with it since and in every step of their maintenance it is guaranteed that the system being modified is consistent. This is performed using three main mechanisms: evolutionary actions, restrictions and change propagation. The evolutionary actions are initiated by the author of the system in order to change it, and are only executed if a set of restrictions holds, which in turn assures that the system will not get to an unusable state. The change propagation mechanism adjusts the whole system to a change, in order to keep it globally consistent: for example, if a concept is removed in the memorization subsystem, the change is propagated into the memorization subsystem itself by removing the associations to that concept, and into the other subsystems by also removing the concept and other elements that reference it. For that taking care of evolution we define a meta level, called the Metasystem, which receives changes from the author and applies them properly or rejects them if they can not be performed.

In the following sections we describe with more detail the elements of the SEM-HP model as they have been implemented in the JSEM-HP tool.

2.1 Memorization Subsystem

The Memorization Subsystem stores, structures, and maintains the contents that will be offered to the students in the form of a global conceptual map that is modeled as a conceptual structure of memorization (CSm).

The CSm is a semantic network with two kinds of nodes: concepts and items, and two kinds of associations: conceptual and functional associations. Concepts are semantically labeled ideas, and items represent the pieces of information

offered to the student, and can be a text document, an image, a video, etc. There is a special concept, called the root concept, which is the central element of the CSm. Conceptual associations connect concepts and have an associated conceptual relation (which is identified with a label, such as 'part of'). Functional associations connect concepts and items, defining the role the item has in relation to the concept (i.e. definition, example, explanation, etc). Figure 1 shows an example of CSm being edited with JSEM-HP, whose root concept is 'OO Concepts'. The main window shows the CSm, in which concepts are represented with ovals, items with rectangles, conceptual relations with arrows and functional associations with lines. The figure also shows two of the windows that can be opened to edit the elements in the CSm. The first one is for editing an item, which has, among other attributes, one URL which points to the document represented by the item. The smallest one shows a list that presents the available functional associations, which can be modified by the author to add more types of associations.

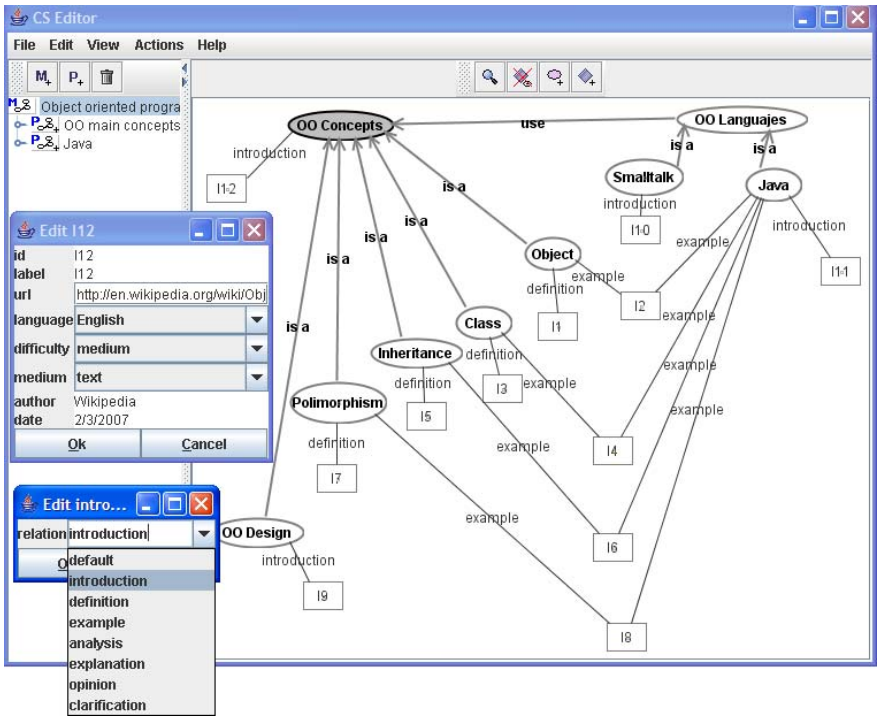


Fig. 1. Example conceptual structure of memorization

2.2 Presentation Subsystem

The Presentation Subsystem allows the author to select subsets of a conceptual structure of memorization in order to create more focused and simpler conceptual maps, which are called conceptual structures of presentation (CSp). The subset is created by hiding or showing elements in the CSm in which it is based. In addition, this subsystem allows the author to change the presentation by modifying the layout of the original CSm, so it suits better to the new CSp. Figure 2 shows two possible presentations that can he defined based on the CSm in figure 1.

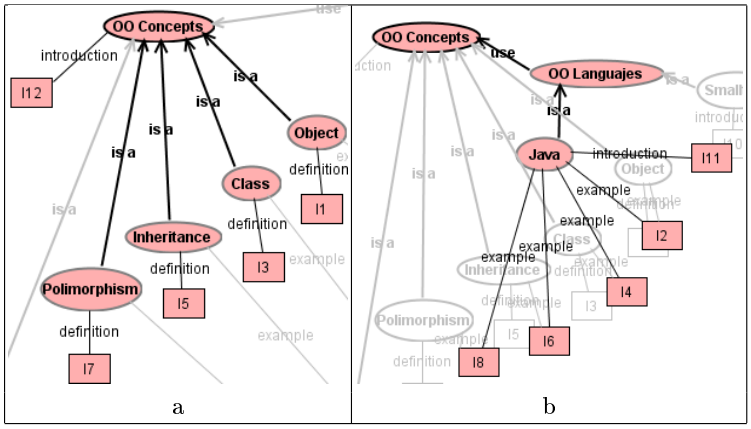


Fig. 2. Two possible presentations

2.3 Learning Subsystem

The Learning Subsystem is in charge of user adaptation. Its main elements are the user model and knowledge and update rules. A set of update and knowledge rules associated to a CSp define a conceptual structure of learning (CSl).

The user model stores the user's features that will be used for adaptation. Among other information, it keeps the user's knowledge degree about the items offered by the system. This is represented with one of these semantic labels: null, low, medium, high and total, each of which has an associated numerical value ranging from 0 to 4.

Knowledge rules define the minimum knowledge that the student needs in order to be ready to understand an item. For example, figure 3 shows two possible knowledge rules for the item I5 in figure 2a, each of which defines one way in which the student can get ready to understand I5 (by having at least a medium knowledge about I12, or about I1 and I3).

Update rules model how the student's knowledge increases while he navigates the conceptual map. Each item has an associated update rule, which has a set

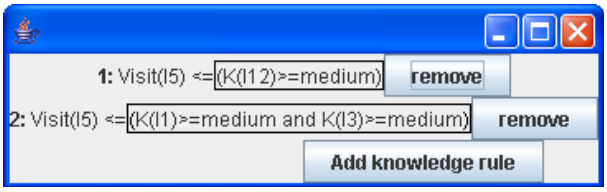


Fig. 3. Example Knowledge rules

of update predicates that are fired when the user visits the item. Each predicate defines a knowledge update about an item, so the author can define a knowledge update about the visited item, but also about other related items. The update of the user's knowledge about an item can be incremental (the user's knowledge is increased), or fixed (it is set to a given degree); absolute (the predicate uses a specific semantic label) or relative (it is based on the degree of user knowledge about the visited item); and first-time (performed only first time the item in the head of the rule is visited) or each-time (it is run every time that the head item is visited). Figure 4 shows a possible update rule for the item I12 in figure 2a, which increases the student's knowledge about I12 in two levels every time it is visited, as well as setting the student's knowledge about I1 and I3 to medium, since the author considers that by reading the introduction to object oriented programming concepts (I12), the student also gains medium knowledge about the definitions of object and class (I1 and I3).

Fig. 4. Example update rule

2.4 Navigation by the User

Figure 5 shows a browsing session using the conceptual structure of learning defined in figures 2a, 3 and 4. The left window shows the conceptual map that is being navigated, the right window is a web browser that shows the currently selected item (I12) and the lower window, which is normally hidden, shows the current student's knowledge state (after visiting I12 twice). In the conceptual map, the items that the student is not ready to understand because his knowledge state does not satisfy any of its knowledge rules are disabled and partially hidden (this is the case of I7). In addition, the concepts and items are annotated with a darker color as the student's knowledge about them increases, so he is aware of his learning process.

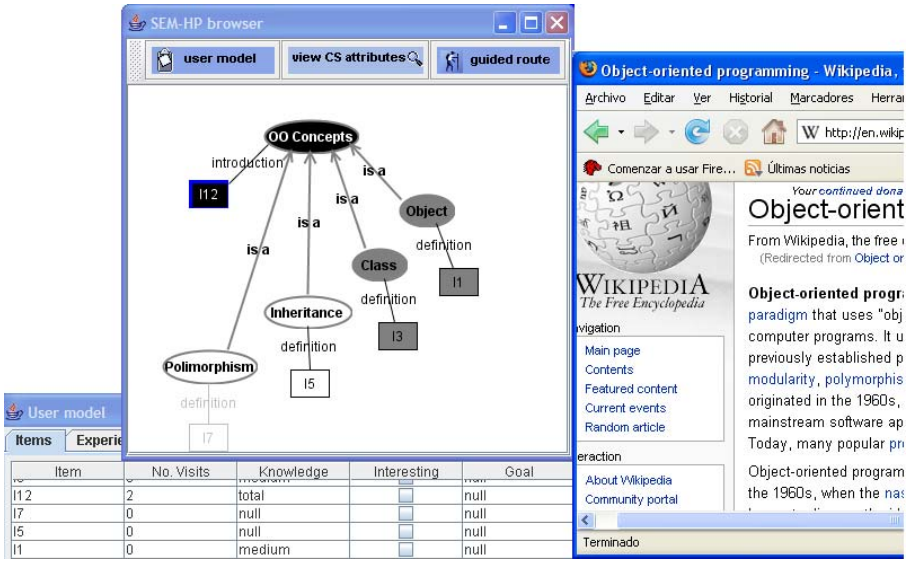


Fig. 5. Navigation by the user

3 Case Example

We have started to use JSEM-HP to aid in our teaching of the subject "Object Oriented Programing" that is taught (in Spanish) at the University of Granada. Different concept maps are used in the initial lectures to introduce the main concepts, and in the subject's web site in order to ease the navigation through the web's contents. These maps were initially created and exported to html with Microsoft Word, which caused problems with non-Microsoft's web browsers and made changes to existing maps difficult. JSEM-HP does not have these problems, so we decided to use the tool to redefine these concept maps, so at a first stage we could test SEM-HP's capabilities in knowledge representation.

Figure 6 shows the initial concept map (in Spanish), and figure 7 shows SEM-HP's version, translated to English. We can see that they are equivalent, aside from: a) the concepts Students, Exams and Theory are associated in JSEM-HP with three binary associations, while in the original map they are joined by a single tertiary association (meaning that students take theory exams), because JSEM-HP does not support tertiary associations; and b) the original map shows an isolated concept that is converted to the Help concept in the new map, which is connected to the root Object Oriented Programming concept, because SEM-HP does not allow unconnected concepts in order to keep consistency.

The result of this experience was that the original author preferred SEM-HP's version, since the lack of tertiary associations could be solved well for him with binary associations, and the ease of changing the already created maps and JSEM-HP's compatibility with any browser were important benefits for him. Being forced by the tool to connect all the concepts was not considered intrusive

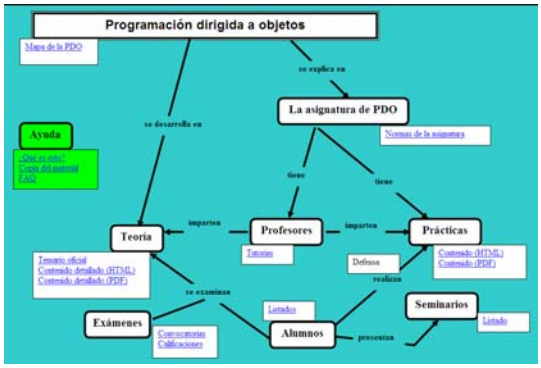


Fig. 6. Initial concept map done with Microsoft Word

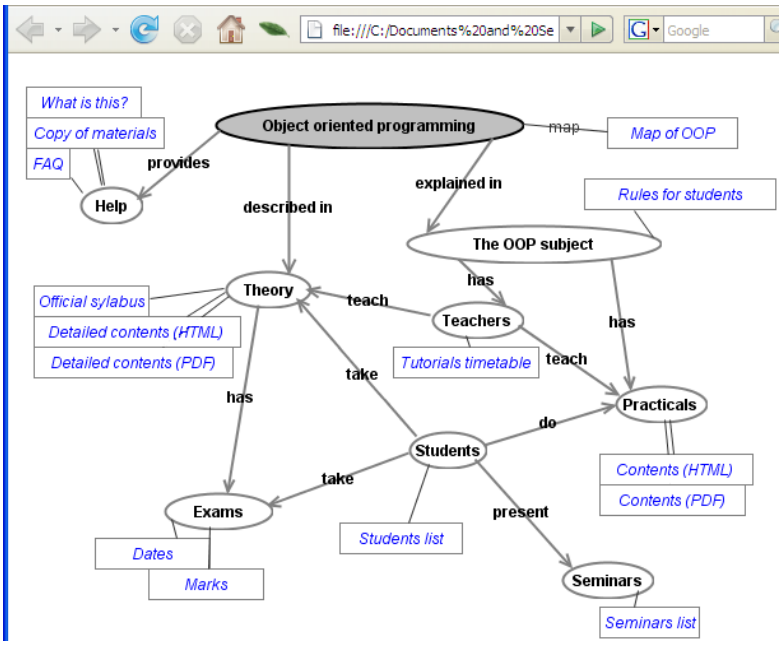


Fig. 7. Current concept map done with JSEM-HP

(in fact it allowed to create a more consistent concept map), and the explicit definition of the root concept required by JSEM-HP was straightforward for the author. In addition, the author positively valued the possibility of defining knowledge and update rules and using JSEM-HP's adaptive features.

4 Related Work, Conclusions and Further Work

There are two areas in which we can find different related works: conceptual maps and adaptive hypermedia systems. Regarding tools for the development of

conceptual maps, we would like to mention CmapTools [7], which implements navigable concept maps that can also be shared and commented. CmapTools is more mature than JSEM-HP, and the main difference lies in its lack of user adaptation and evolutionary support. Regarding adaptive hypermedia systems, we can mention, among other relevant work, the AHA! architecture [8], based on the AHAM model [9], which employs several adaptive techniques that are similar to SEM-HP's, although they are focused to a more traditional web browsing instead of concept maps. None of the reviewed works, in both fields, employ SEM-HP's evolutionary approach, that guarantees the system's consistency during its development and evolution. To this extent, we believe that SEM-HP's main contribution is grouping its semantic orientation by means of navigable concept maps with its user adaptation and evolutionary capabilities.

Our further work includes expanding the SEM-HP model to allow new elements such as tertiary associations, refining and expanding JSEM-HP to include more features already defined in the model such as feedback [10], and using the tool in more real cases in order to validate its usefulness.

References

1. Novak, J., Gowin, D.: Learning how to learn. Cambridge University Press (1984)
2. Horton, P., McConney, A.A., Gallo, M., Woods, A.L., Senn, G.J., Hamelin, D.: An investigation of the effectiveness of concept mapping as an instructional tool. Science Education **77(1)** (1993) 95–111
3. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction **6**(2-3) (1996) 87–129
4. García-Cabrera, L., Rodríguez-Fórtiz, M.J., Parets-Llorca, J.: Evolving hypermedia systems: a layered software architecture. Journal of Software Maintenance and Evolution: Research and Practice **14(5)** (2002) 389–405
5. Medina-Medina, N., Molina-Ortiz, F., Garcia-Cabrera, L.: Diversity of structures and adaptive methods on an evolutionary hypermedia system. IEE Proceedings - Software **152**(3) (2005) 119–126
6. Molina-Ortiz, F., Medina-Medina, N., García-Cabrera, L.: An author tool based on SEM-HP for the creation and evolution of adaptive hypermedia systems. In: ICWE '06: Workshop proceedings of the sixth international conference on Web engineering, New York, NY, USA, ACM Press (2006)
7. Cañas, A.J., Carff, R., Hill, G., Carvalho, M.M., Arguedas, M., Eskridge, T.C., Lott, J., Carvajal, R.: Concept maps: Integrating knowledge and information visualization. In: Knowledge and Information Visualization. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2005) 205–219
8. De Bra, P., Aerts, A., Berden, B., Lange, B.D., Rousseau, B., Santic, T., Smits, D., Stash, N.: Aha! the adaptive hypermedia architecture. In: Proceedings of the ACM Hypertext Conference, Ottingham, UK. (2003)
9. Wu, H., Houben, G., Bra, P.D.: Aham: A reference model to support adaptive hypermedia authoring. In: Proceedings of the Conference on Information Science, Antwerp. (1998) 51–76
10. Medina-Medina, N., Molina-Ortiz, F., García-Cabrera, L.: An adaptation method by feedback in an evolutionary hypermedia system. In: ICWE '06: Proceedings of the 6th international conference on Web engineering, New York, NY, USA, ACM Press (2006) 161–168

JSEM-HP: A TOOL FOR THE DEVELOPMENT OF ADAPTIVE EVOLUTIONARY HYPERMEDIA SYSTEMS

Fernando Molina Ortiz, Nuria Medina Medina
Universidad de Granada
Depto. de Lenguajes y Sistemas Informáticos. E.T.S. de Ingeniería Informática.
C./ Periodista Daniel Saucedo Aranda s/n. 18071 - Granada. Spain

Lina García Cabrera
Universidad de Jaén
Depto de Informática. Escuela Politécnica Superior.
Campus Las Lagunillas. 23071 – Jaén. Spain

ABSTRACT

This paper presents JSEM-HP, a tool for the development of hypermedia systems. It is based on the SEM-HP model, which adds semantics, user adaptation and evolution to traditional hypermedia systems. The hypermedia systems created with this tool are based on a special kind of semantic network, which gives semantic coherence to the information offered to the user. The conceptual structure that is offered to the user is dynamically adapted to the user knowledge, so he will not access items he is not ready to understand, and will have awareness of his process of learning. In addition, the tool supports the evolution of the system created with it, assuring that changes made to a system keep it in a consistent state.

KEYWORDS

Hypermedia systems, user adaptation, software evolution.

1. INTRODUCTION

Hypermedia systems can be seen as a collection of nodes, which have the information offered to the user, and links, which can be followed by the user in order to navigate to another node. While these systems have several advantages, such as being able to provide a non-linear access, there are also disadvantages: they can lead to disorientation ("getting lost" in the network of links), and knowledge overhead (caused by the additional effort of knowing where in the network the user is and which link to follow next). Adaptive hypermedia systems [Brusilovsky, 1996] address these problems by adapting the information and/or the link structure to the user: if the offered information and link structure takes into account what the current user already knows, his goals, etc., it will be easier for him to navigate due to the reduction of disorientation and knowledge overhead. In addition, a hypermedia system can grow to be a complex software system, so we should consider applying software engineering resources to assure that we properly build our hypermedia system. Software evolution understands that software is always evolving, and we must assure that this evolution keeps the system in a proper state.

In this paper we present a tool based on the SEM-HP model [García-Cabrera et al, 2002] [Medina-Medina et al, 2005], which allows the development of semantic, adaptive and evolutionary hypermedia systems. This model takes into account the evolution of the hypermedia system, in order to assure that it is done properly, and it also addresses the disorientation and cognitive overhead problems by applying several adaptive techniques. Firstly we introduce the SEM-HP model, further describing it while we show JSEM-HP, a tool based on it that implements the key elements of the model. Finally, related work, conclusions and further work are shown.

2. THE MODEL SEM-HP

SEM-HP is a semantic, systemic and evolutionary model for the development of adaptive hypermedia systems. It is semantic because the information it provides is semantically structured by means of special semantic networks we call conceptual structures.

It is systemic because it conceives a hypermedia system as composed by four interrelated subsystems: memorization, presentation, navigation and learning. The memorization subsystem allows the author to build the conceptual and informational domains that comprise the elements that can be offered to the user. The presentation subsystem permits to select different presentations or subsets of the contents defined in the memorization subsystem. The navigation subsystem provides tools to decide the order in which the information can be browsed according to the associations between concepts, and the learning subsystem takes care of the user modeling and user adaptation, directing the user navigation in function of his knowledge. In this paper we will focus in the memorization, presentation and learning subsystems.

In order to assist the evolution of the system, the model provides four elements: evolutionary actions, metasystem, restrictions and change propagation. The system is created and evolved by means of evolutionary actions, which are initiated by the author through a user interface, and sent to the metasystem. By checking a set of restrictions, the metasystem decides if the execution of the action is possible (it would leave the system in a consistent state), and runs it if it is so, therefore modifying the system. Some changes need to be propagated so consistency is kept both in the same subsystem and among subsystems, so the metasystem also takes care of change propagation (for example, when an element is removed in the memorization subsystem, this element and the references to it must also be removed in the other subsystems).

3. THE TOOL JSEM-HP

JSEM-HP is a tool that implements the SEM-HP model. It is written in Java due to its portability, the availability of plenty of third-party libraries, and the facilities available for its use in Internet. It is based only in free software (mainly in JGraph [JGraph] for graph manipulation), and it is planned to be available as free software. It allows the creation of hypermedia systems as defined in SEM-HP, and uses a software subsystem for each one of the subsystem defined in SEM-HP. In the following subsections we describe the tool, along with the aspects of SEM-HP it implements.

3.1 Memorization Subsystem

The main element in the memorization subsystem is the conceptual structure of memorization (CSm). The CSm is a semantic network with two kinds of nodes: concepts and items. Concepts are ideas labeled semantically and items are the pieces of information offered by the system. The concepts can be associated by conceptual associations, and items are connected to concepts by means of functional associations, which describe the role the item has in describing the concept (definition, example, explanation, etc). In figure 1 we show a CSm as it can be created with the tool.

Concepts are represented with ellipses and items with rectangles, conceptual associations with arrows and functional associations with lines. The figure also shows the attributes associated to an item (‘Edit I1’ window), which include a URL that will be used by the browser to show the information in the item, which can be text, a web page, a graphic, etc. In addition, we show the different roles available for a functional association (‘Edit explanation’ window), which can be extended by the author.

Besides the semantic network, the CSm also contains a set of weight rules (Rw). There is a weight rule associated to every concept in the CSm, and it has a weight for each item connected to it. Each weight represents the importance of its item in describing the concept, and in the tool it is a number from 0 to 100. By default all the items associated to a concept have the same weight, but this can be changed by the author. Weight rules are used to calculate the user’s knowledge about a concept from his knowledge about the items associated to it. Figure 1 shows in the bottom-right the weights given to the items associated to the concept ‘Granada’.

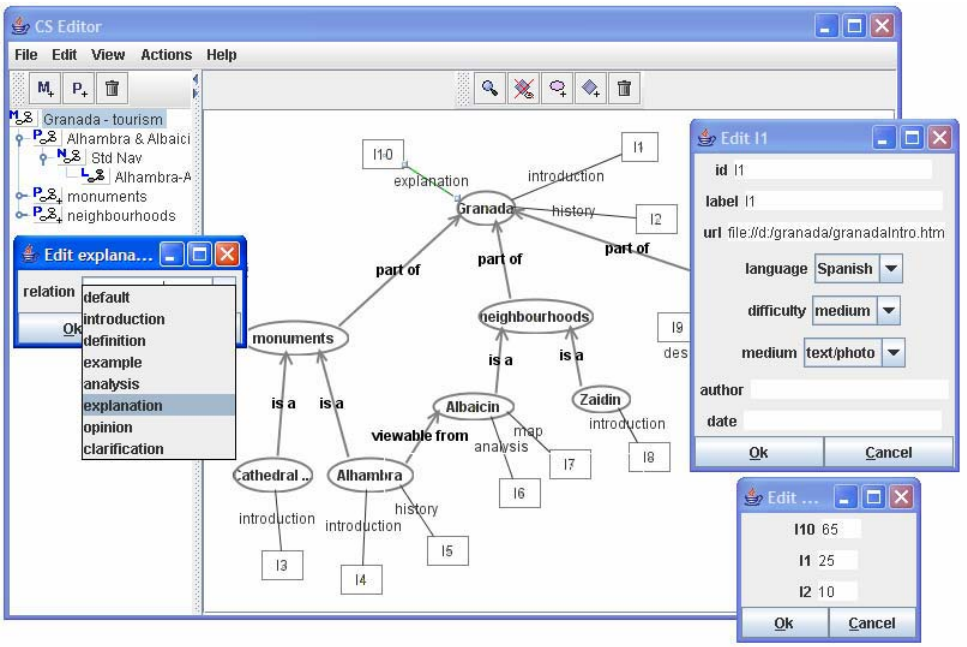


Figure 1. Example CSm in the memorization subsystem

3.2 Presentation Subsystem

The presentation subsystem allows the author to select different subsets of a CSm, creating different conceptual structures of presentation (CSp). Each CSp represents a different knowledge subdomain of its CSm. In addition, the author can spatially rearrange and resize the elements of the CSm he adds to the CSp, defining a different way to present them to the user.

3.3 Learning Subsystem

The learning subsystem takes care of user adaptation. The main elements in the learning subsystem are the user model, the update rules (Ru) and the knowledge rules (Rk).

The *user model* represents the user in the system, and it contains the information about him that is used for adaptation, which includes his knowledge, experience, preferences and interests. The user knowledge about every item in the CSm is represented with one of the following semantic labels: null, low, medium, high and total, each of which has an associated numerical value ranging from 0 to 4).

Update rules are used to update the user knowledge while he browses the system. There is an update rule for every item in the conceptual structure, which is activated when the user reads the item, and updates the user knowledge about the visited item, and possibly others. The structure of an update rule is as follows:

$$\text{Visit}(I) \rightarrow \text{Update}(I), \text{Update}(I_1) \ldots \text{Update}(I_n)$$

Where the head of the rule is the visited item and the body includes a set of update predicates that will be executed after the visit. The update of the user knowledge about an item can be: incremental (the degree of knowledge is increased) or fixed (the degree of knowledge is set to a fixed degree of knowledge), absolute (it uses an absolute semantic label) or relative (it uses the degree of user knowledge about the visited item), first-time (it is ran only the first time the item in the head is visited) or each-time (it is ran every time that the head item is visited), providing eight different update predicates. For example, the following rule could be used for the item I1 in figure 1:

Visit(I1) → Inc-abs(I1,3), Fix-abs(I2,low), Fix-abs(I10,low)

The first predicate increases the knowledge about I1 in three steps every time the user visits the item. Therefore, in order to obtain total knowledge about I1, the user must visit it twice. The second and third predicates set the user knowledge about I2 and I10 (history and explanation of Granada) to low, since the item I1 contains some information about them and the author has considered that by visiting I1 the user also gets some knowledge about I2 and I10.

Knowledge rules are used to define the knowledge prerequisites needed in order to access an item. In this way, the author can define one or more knowledge rules for each item, one of which must be satisfied by the user knowledge state if he wants to access it. The structure of an Rk is as follows:

Knowledge_restriction (I_1) opL…opL Knowledge_restriction (I_n) → Visitable(I)

Where opL is a logic operator (And or Or), and Knowledge_restriction(I) is a knowledge restriction about the item I that the user must satisfy. A knowledge rule for the item I5 in figure 1 could be the following:

K(I2) >= low And K(I4) >= medium → Visitable(I5)

3.4 Navigation by the User

In SEM-HP, the user browses a conceptual structure directly, selecting the items in which he is interested. When the user clicks an accessible item the system opens a web browser window which shows the contents of the selected item. The conceptual structure is adapted to the user by disabling the items for which the knowledge rules are unsatisfied by the user's current knowledge, thereby providing the user only with the items he is ready to understand. In addition, the conceptual structure is annotated in order to allow the user to see his estimated knowledge about the elements in the conceptual structure. Figure 2 shows a conceptual structure based on the one in figure 1 while navigated by the user, after he clicks in I1. A web browser is opened to show the item's contents, and I1, I2 and I10 are updated using their update rules. The figure shows the calculated knowledge about the items, which is annotated in a darker color as the user knowledge increases (white=null, black=total), as it happens with the concept Granada, for which the knowledge is calculated according to the weight rule in figure 1. Items I5 and I6 are hidden (drawn in light grey) and disabled (selecting them has no effect) because their knowledge rules are not satisfied by the user.

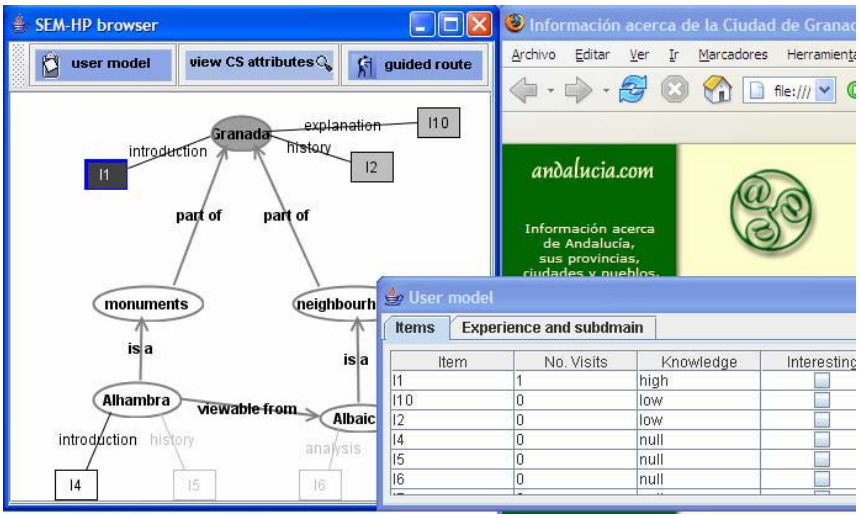


Figure 2. Navigation by the user

4. RELATED WORK, CONCLUSIONS AND FURTHER WORK

There are several tools that are conceptually similar to JSEM-HP, specifically regarding to the usage of a knowledge representation model for semantically structuring and offering hypermedia information. These include Concilla [Nilsson et al, 1999] and CmapTools [Cañas et al, 2004], both of which are still used and in further development. In this section we will focus on the latter, which is a set of tools that allow building, navigating, sharing and criticizing knowledge models. CmapTools is based on conceptual maps (which contain associated concepts forming propositions), and it allows the elements in a map to be associated to resources that complement the information offered by the map and can be web pages, sounds, etc. These maps are similar to our conceptual structures with concepts, associations, and items (resources). One difference lies in the treatment of items: while CmapTools groups them according to their type (pdf, image, etc.), SEM-HP uses roles, which are semantically more adequate in our opinion.

CmapTools exceeds JSEM-HP in the collaborative development and sharing of conceptual maps, for which includes a network of CmapTools servers in Internet. In addition, CmapTools has a wide user base and has been used in multiple environments while JSEM-HP is starting to be used.

Although CmapTools is based in well established knowledge representation models, it lacks the formal groundings that lie beneath JSEM-HP regarding the evolution of hypermedia systems. In addition, JSEM-HP has other features that are not found in this and other related approaches, such as user adaptation.

SEM-HP groups several desirable features in a hypermedia system, both from the development and usage points of view. Firstly, its evolutionary approach allows the evolution of the hypermedia system while maintaining the desired consistency. Secondly, the semantic focus ensures that the information offered is given a semantic coherence, which is translated to the user via a navigable conceptual structure. Finally, the system can adapt to the user while he navigates, reducing the problems of disorientation and cognitive overhead, and making the user aware of his progress.

Although the development of hypermedia systems with the tool ease the author's work by rejecting incorrect actions and keeping the system consistent by automatically propagating changes, if the author wants to take advantage of the user adaptation features, he must learn to use and define the proper weight, update and knowledge rules. This additional work for the author might limit the success of the tool, since he might not be willing to learn a new tool that can seem complex to a non computer scientist. In this sense the update and knowledge rules have been simplified, and a new and more intuitive user interface is being built.

The future work includes the further development of the tool to include features of SEM-HP not mentioned in this paper, such as guided routes ([Medina-Medina et al 2003]), other navigation modes, interesting items, feedback, etc [Medina-Medina, 2004], and to use it in real world applications, in order to validate its usefulness and to correct possible problems.

REFERENCES

Brusilovsky P., 1996. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6: 87-129. Kluwer Academic Publishers.

Cañas, A. J. et al, 2004. CmapTools: A Knowledge Modeling and Sharing Environment. *Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping*, Universidad Pública de Navarra: Pamplona, Spain. pp. 125-133.

García-Cabrera, L. et al, 2002. Evolving hypermedia systems: a layered software architecture. *Journal of Software Maintenance and Evolution: Research and Practice.* John Wiley & Sons, Ltd. 14(5). 389-405

JGraph. *http://www.jgraph.com.*

Medina-Medina, N. et al, 2003. Personalized Guided Routes in an Adaptive Evolutionary Hypermedia System. *EUROCAST 2003. Lecture Notes in Computer Science 2809. Springer 2003.* ISBN 3-540-20221-8. pp. 196-207.

Medina-Medina N., 2004. An integral evolutionary model of adaptation for hypermedia systems: SEM-HP's learning system (in Spanish). *PhD Thesis*. University of Granada.

Medina-Medina, N. et al, 2005. Diversity of structures and adaptive methods on an evolutionary hypermedia system. *IEE Proceedings – Software.* Vol 152, Issue 3. 119-126.

Nilsson, M. et al, 1999. Conzilla - Towards a concept browser. CID-53, TRITA-NA-D9911, Department of Numerical Analysis and Computer Science, KTH, Stockholm.

An Author Tool Based on SEM-HP for the Creation and Evolution of Adaptive Hypermedia Systems

Fernando Molina-Ortiz
University of Granada
E.T.S.I.I., Dept. LSI, 31
Granada 18071. Spain
(+34) 958 243180
fmo@ugr.es

Nuria Medina-Medina
University of Granada
E.T.S.I.I., Dept. LSI, 25
Granada 18071. Spain
(+34) 958 240634
nmedina@ugr.es

Lina García-Cabrera
University of Jaén
Campus Las Lagunillas, A-3
23071 - Jaén. Spain
(+34) 953 212475
lina@ujaen.es

ABSTRACT

This paper presents JSEM-HP, an author tool for the development of semantic, adaptive and evolutionary hypermedia systems based on the model SEM-HP. The proposed development process has four iterative phases, associated to four interacting subsystems: memorization, in which the author creates a specific kind of semantic network that gives semantic coherence to the information offered to the user; presentation, which allows to select different subsets of the semantic net; navigation, which permits to define an order to navigate the items according to their semantic relations; and learning, which takes care of user modeling and adaptation. The tool supports the evolution of the systems created with it, assuring that changes made in a subsystem keep this subsystem and the whole system in a consistent state. The user adaptation supported by the tool allows the semantic net that is offered to the user to be dynamically adapted to the user knowledge, so he will not access the information he is not ready to understand, and he will have awareness of his process of learning.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia

General Terms

Design, Human Factors

Keywords

Hypermedia Systems, User Adaptation, Software Evolution

1. INTRODUCTION

Hypermedia systems have proved to be a very useful way of structuring and displaying information, specially since their usage in the World Wide Web. While these systems have several advantages, such as being able to provide a non-linear access, there are also disadvantages. Among the disadvantages, the most important are disorientation, which arises when the user gets lost in the network of links, and knowledge overhead, caused by the additional effort of knowing where in the network the user is and which link to follow next. Adaptive hypermedia systems [1] address these problems by adapting the information and/or the link structure to the user, ideally taking into account aspects such as what the current user already knows, his goals, preferences, profile, etc. Techniques such as reducing the number of available links or ordering them according to the user, not permitting the access to information the user is not ready to understand or providing additional information that will help the user to know his situation into the hypermedia system will make the navigation easier and more productive due to the reduction of disorientation and knowledge overhead.

In addition, a hypermedia system can grow to be a complex software system, so we should consider applying software engineering resources to assure that we properly build our hypermedia system. Software is always evolving, and we must assure that this evolution keeps the system in a suitable state. This is done on the basis of the work presented in [14], as described in [6].

In this paper we present a tool based on the SEM-HP model [6, 9] that allows the development of semantic, adaptive and evolutionary hypermedia systems. It takes into account the evolution of the hypermedia system order to assure that it is done properly, and it also addresses the disorientation and cognitive overhead problems by applying several adaptive techniques. The paper is structured as follows: Firstly we introduce the SEM-HP model (section 2), further describing it while we show JSEM-HP, a tool based on it that implements the key elements of the model (section 3). In section 4 we describe some design aspects of JSEM-HP, before discussing related work in section 5. Finally, conclusions and further work are presented (section 6).

2. THE MODEL SEM-HP

SEM-HP is a SEMantic, Systemic and Evolutionary Model for the development of adaptive HyPermedia systems.

It is semantic because the information it provides is semantically structured by means of what we call conceptual structures, which are a specific kind of semantic network.

It is systemic because it conceives a hypermedia system as composed by four interrelated subsystems: memorization,

ICWE'06 Workshops, July 10-14, 2006, Palo Alto, CA
ACM 1-59593-435-9/06/07

presentation, navigation and learning. The memorization subsystem allows the author to build the conceptual and informational domains that comprise the elements that can be offered to the user. The presentation subsystem permits to select different presentations or subsets of the contents defined in the memorization subsystem. The navigation subsystem provides tools to decide the order in which the information can be browsed, and the learning subsystem takes care of the user modeling and user adaptation, therefore allowing the development of adaptive hypermedia systems.

The model is evolutionary because it takes care of the evolution of the system. This is done through four elements: evolutionary actions, metasystem, restrictions and change propagation. The system is created and evolved by means of evolutionary actions, which are initiated by the author through a user interface, and sent to the metasystem. By checking a set of restrictions, the metasystem decides if the execution of the action is possible (it would leave the system in a usable and consistent state), and runs it if it is so, therefore modifying the system. Some changes need to be propagated so consistency is kept both in the same subsystem (internal propagation) and among subsystems (external propagation), so the metasystem also takes care of change propagation. For example, when an element is removed in the memorization subsystem, this element and the references to it must also be removed in the other subsystems.

In Figure 1 we show the architecture proposed by SEM-HP, which defines a double division: a vertical division between system and metasystem and a horizontal division between the four subsystems.

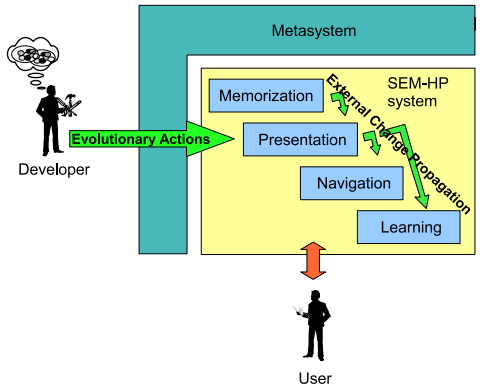


Figure 1: SEM-HP architecture

In addition SEM-HP proposes a development process with four iterative phases, each one corresponding to one of the subsystems, as shown in Figure 2.

The central navigational structure in SEM-HP is the conceptual structure, which is a specific kind of semantic network in which the user can click in order browse it. The model proposes four different navigation modes, which vary depending on whether the user navigation is restricted or not, how it is restricted, and the way the conceptual structure and the information in the items is presented. In this paper we will focus in the navigation restricted by knowledge in which, among other things, the system only allows

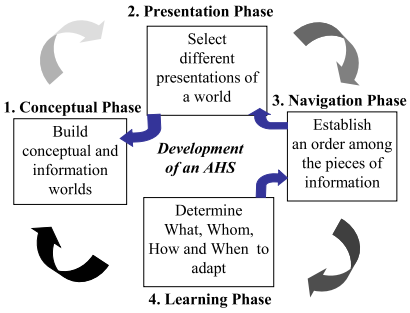


Figure 2: SEM-HP development process

the user to access the elements that he is ready to understand.

3. THE TOOL JSEM-HP

JSEM-HP (Java-based SEM-HP) is a tool that implements the JSEM-HP model. It follows the development process defined in SEM-HP, allowing the creation of hypermedia systems as defined in SEM-HP. In the following subsections we describe the tool, along with the aspects of SEM-HP it implements.

3.1 Memorization Subsystem

This subsystem is associated to the Conceptual Phase in Figure 2. The main element in the memorization subsystem is the conceptual structure of memorization (CS_M). The CS_M is a semantic network with two kinds of nodes: concepts and items. Concepts are ideas labeled semantically and items are the pieces of information offered by the system. The concepts can be associated among them by labelled conceptual associations, and items are connected to concepts by means of functional associations, which describe the role the item has in describing the concept (definition, example, explanation, etc). In Figure 3 we show a CS_M as it can be created with the tool.

Concepts are represented with ellipses and items with rectangles, conceptual associations with arrows and functional associations with lines. Figure 3 also shows the attributes associated to an item (Edit I1 window), which include a URL that will be used by the browser to show the information in the item, which can be text, a web page, a graphic, etc. In addition, we show the different roles available for a functional association (Edit explanation window). The tool provides an initial set of available roles, but the author is allowed to add new ones if he thinks it is necessary.

Besides the semantic network, the CS_M also contains a set of weight rules. There is a weight rule associated to every concept in the CS_M, which has a weight for each item connected to it. Each weight represents the importance of its item in describing the concept, and in the tool it is a number from 0 to 100. The sum of the weights of all the items connected to a concept must be 100. By default all the items associated to a concept have the same weight, but this can be changed by the author. Weight rules are used to calculate the user's knowledge about a concept from his

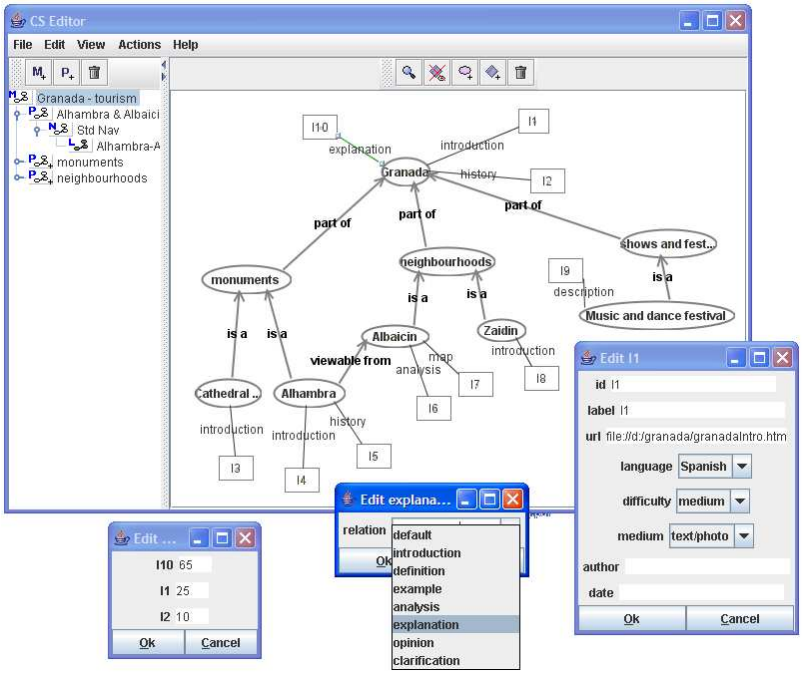


Figure 3: Example CS_M in the memorization subsystem

knowledge about the items associated to it. Figure 3 shows in the bottom-left the weights given to the items associated to the concept Granada.

The metasystem assures that there are no inconsistencies in the conceptual structure, such as unconnected concepts, more than one concept with the same name, weight rules whose weights do not add to 100, etc., some times by rejecting actions and others by propagating changes.

3.2 Presentation Subsystem.

The presentation subsystem allows the author to select different subsets of a CS_M, creating different conceptual structures of presentation (CS_P). Each CS_P represents a different knowledge subdomain of its CS_M. In addition, the author can spatially rearrange and resize the elements of the CS_M he adds to the CS_P, defining a different way to present them to the user. The author gives two attributes to each CS_P: its name and the knowledge subdomain it represents. Figure 4 shows two different presentations that can be created from the previous CS_M. The tool shows in light grey the elements in the CS_M that are currently hidden, and in black with coloured background the elements already included. The metasystem ensures that each CS_P is a consistent conceptual structure (it has no unconnected elements, all its elements are present in its CS_M, etc).

3.3 Navigation Subsystem

This subsystem is mainly used for the mode of navigation called navigation by conceptual relation, which is out of the scope of this paper. The main idea is to restrict the user's navigation in a way that is consistent with the conceptual relationships in the conceptual structures. In order to do this, the system creates a default set of order rules, based on temporal logic, that allow the user to visit an item after another only if he follows a conceptual relation by visiting items associated to connected concepts. The author is also given the option to modify the default order rules by adding other prerequisites which are only based on the user's navigation history. There is more information about this subsystem in [6].

3.4 Learning Subsystem

The learning subsystem takes care of user adaptation. The main elements in the learning subsystem are the user model, the update rules and the knowledge rules, which will be discussed bellow. The learning subsystem allows the author to create different conceptual structures of learning (CS_L), based on a conceptual structure of navigation, each of which can be built specifically for certain kind of user. For example, a more restrictive CS_L could be defined for users with little experience in navigation so they are more tightly guided and chances for them to get lost while navigating are reduced. Figure 5 shows the attributes that are used to label a CS_L (the knowledge subdomain is inherited from the presentation it was defined from). The system will automatically choose the CS_L that is most appropriate for its user, by searching the one that best matches his profile.

3.4.1 User Model

The user model represents the user in the system, and it

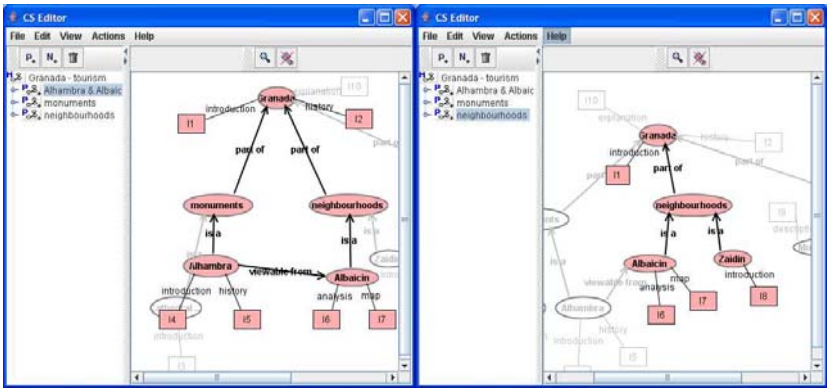


Figure 4: Two possible presentations

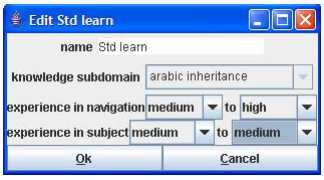


Figure 5: Attributes of a CS_L

contains the information about him that is used for adaptation, which includes his knowledge, experience, preferences and interests. The parts of the user model we will mention here are the user knowledge and his knowledge goal. The user knowledge about every item in the CS_M is represented with one of the following semantic labels: null, low, medium, high and total; each of which has an associated numerical value ranging from 0 to 4. This knowledge is automatically calculated while the user navigates, as we will show later. The user's knowledge goal is the degree of knowledge he wishes to achieve about certain items in the conceptual structure, and modeled as a set of pairs (item, semantic label). Figure 9 shows a view of part of the user model, which includes his knowledge state and goal.

3.4.2 *Update Rules*

Update rules are used to update the user knowledge while he browses the system. There is an update rule for every item in the conceptual structure, which is activated when the user visits the item, and updates the user knowledge about the visited item, and possibly others. The structure of an update rule is shown in (1).

$$Visit(I) \rightarrow Update(I), Update(I_i)...Update(I_n) \quad (1)$$

Where the head of the rule is the visited item and the body includes a set of update predicates that will be executed after the visit. The update of the user knowledge about an item can be incremental (the user knowledge is increased), or fixed (it is set to a given degree); absolute (the predicate uses a specific semantic label) or relative (it is based on the degree of user knowledge about the visited item); and first-time (performed only first time the item in the head of the rule is visited) or each-time (it is run every time that the head item is visited).

The combination of these features provide eight different update predicates. Figure 6 shows an example of update rule being edited for the item I1 in the conceptual structure used before, that reads as: Visit(I1)→ Inc-abs(I1,3), Fix-abs(I2,low), Inc-rel(I10,-2). The first Inc-abs predicate increases the knowledge about I1 in three steps every time the user visits the item. Therefore, in order to obtain total knowledge about I1, the user must visit it twice. The second predicate sets the user knowledge about I2 (history of Granada) to low, since the item I1 contains some information about the history of Granada and the author has considered that by visiting I1 the user also gets some knowledge about I2. The third predicate sets the knowledge about I10 to the user knowledge about I1 (the head of the rule) minus two. Therefore, after the user visits I1 for the first time his knowledge about I1 will be set to 3 (high), his knowledge about I2 will be set to low, and his knowledge about I10 to 3 minus 2, which is 1 (low). After the second visit to I1 his knowledge about I1 will be total, and about I10 it will be medium. The figure also shows the available update predicates, as provided by the tool.

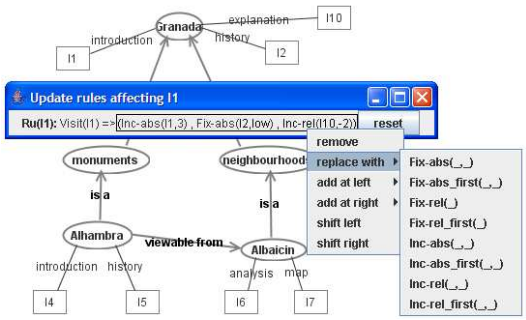


Figure 6: Example update rule

3.4.3 Knowledge Rules

Knowledge rules are used to define the knowledge prerequisites needed in order to access an item. In this way, the author can define one or more knowledge rules for each item, one of which must be satisfied by the users knowledge state if he wants to access it. The structure of a knowledge rule is shown in (2).

$$KRestr(I_1)\, opL \ldots opL\, KRestr(I_1) \rightarrow Visitable(I) \quad (2)$$

Where opL is a logic operator (And or Or), and KRestr(I) is a knowledge restriction about the item I that the user must satisfy. Figure 7 shows two possible knowledge rules being edited for the item I5. The first rule (Visit(I5)← K(I2)≥low and K(I4)≥low) states that, in order to access the history of the Alhambra (I5), the user must have a knowledge of low or higher about both the history of Granada (I2) and the introduction to the Alhambra (I4). The second rule (Visit(I5)← K(I10) ¿ medium) says that another way of accessing the item is by having a knowledge larger than medium about the explanation on Granada in I10. The figure also shows the knowledge restrictions provided by the tool.

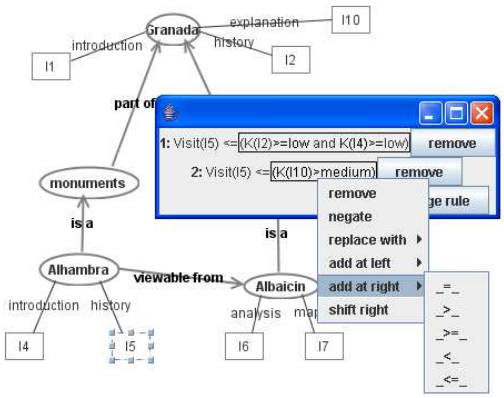


Figure 7: Example knowledge rules

We would like to point out that update and knowledge rules are created using a graphic editor, so the author does not need to write the text associated to the rules, and it is ensured that correct rules are entered.

The metasystem guarantees that the elements in the learning subsystem are consistent and that the update and knowledge rules allow that all the items will eventually become visitable as the user browses the system, and that the user can gain total knowledge about every item in the conceptual structure [8].

3.5 Tree of Conceptual Structures

As we have seen, several CS_P can be created based on the same CS_M, different CS_N can be created based on a CS_P, and several CS_L can be created based on a CS_N, producing a tree-like structure in which the root is a CS_M. The JSEM-HP tool provides a left panel which can be used by the author in order to manage the tree of CS he is working with, as it can be seen in Figure 8.

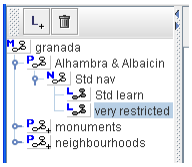


Figure 8: Example of a tree of conceptual structures

3.6 Navigation by the User

SEM-HP allows for many adaptation methods [1], including guidance and adaptive presentation and navigation [9, 10]. For now, JSEM-HP implements some of those adaptive techniques, including personalized views (the system choses the most suitable CS_P), map adaptation (the conceptual structure is used for navigation and it is adapted to the user), annotation (the user knowledge is annotated so he is aware of how his knowledge increases), and link hiding and disabling (the items the user is not ready to understand are partially hidden and disabled).

In SEM-HP, the user browses a conceptual structure directly, selecting the items in which he is interested. When the user clicks an accessible item the system opens a browser window which shows the contents of the selected item. The conceptual structure is adapted to the user by disabling the items for which the knowledge rules are unsatisfied by the user's current knowledge, thereby presenting to the user only the items he is ready to understand. In addition, the conceptual structure is annotated in order to allow the user to see his estimated knowledge about the elements in the conceptual structure. Figure 9 illustrates the user interface presented to the user when browsing the conceptual structure built in figures 3, 6 and 7, after he clicks in I1 (the actual user interface will not show the user model window, which is included here to make the user's knowledge state more evident). When the user clicks I1, a web browser is opened to show the item's contents, and I1, I2 and I10 are updated as detailed in section 3.4.2. The figure shows the calculated knowledge about the items in the figure, which is annotated in the conceptual structure (in the corresponding item or concept) by using a darker color as the user knowledge increases (white=null, black=total), as it happens with the concept Granada, for which the knowledge is calculated according to the weight rule in Figure 3. Items I5, I6 and I7 are hidden (drawn in light grey) and disabled (the user can not click them), since their knowledge rules are not satisfied by the user. Nevertheless, the user knows that the disabled items are available and will become accessible when he gains enough knowledge. If the user is interested in a specific set of items, he can set them as interesting, and the system will mark the items he needs to visit so the interesting items become accessible [9].

4. DESIGN ISSUES IN JSEM-HP

The language chosen for the implementation of the tool is Java, due to its portability, the availability of plenty of third-party libraries, and the facilities available for its use in Internet. It is based only in free software, and it is planned to be also available as free software. In this section we describe the main design decisions taken in the development of the tool regarding the evolutionary nature of JSEM-HP.

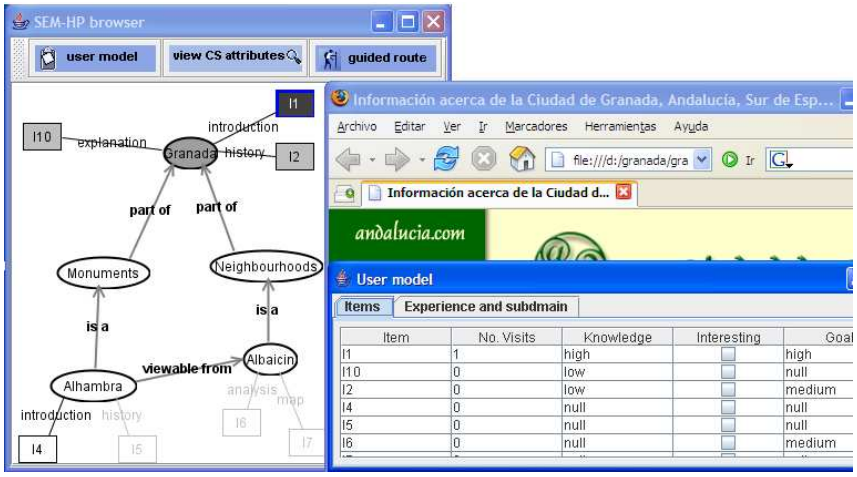


Figure 9: Navigation by the user

4.1 Metasystem and Evolutionary Actions

In order to take care of the evolution of the hypermedia system, an explicit metasystem class has been defined, which runs the evolutionary actions that change any of the subsystems in SEM-HP. Upon the reception of an evolutionary action, the metasystem does the following:

1. Check the action's pre-conditions. If any of them does not hold, the action is rejected.
2. Perform the action.
3. Check the action's post-conditions. If any of them does not hold, the system is brought back to its state previous to the execution of the action, and the action is rejected.
4. Generate the actions that will perform the internal propagation (into the same subsystem), if any, and run them.
5. Generate the actions that will perform the external propagation (into other subsystems), if any. I there will be external propagation, inform the author about the subsystems that will get affected. Run the external propagation actions.

Evolutionary actions are represented by means of classes: for each kind of evolutionary action we have defined a class, being the parameters of the action member variables of the class. An object of the class represents a particular change in a subsystem, which is described by the action itself and its parameters' values. The parameters of an evolutionary action are accessed through a common mechanism that is based in Java's reflexive facilities. Instead of using classes for evolutionary actions we could have decided to simply use methods in the metasystem. We have opted for explicit evolutionary actions as classes because it eases change notification and change propagation (changes are properly described with the action object, which is the same that was sent to perform the modification), and it would allow an easy implementation of an structural history of the system, which would only need a list of evolutionary actions.

The existence of post-conditions allows for an easier definition of some of the evolutionary actions, and it also makes easier to check some restrictions that would be harder to verify before the change (for example, checking that there will still be a path between two concepts after removing a relation is simpler to do after actually removing the relation). This makes necessary the addition of a mechanism that allows to bring the system back to a previous state in case the action is rejected because any post-condition does not hold (this mechanism is necessary anyway in order to provide a nowadays inprescindible undo feature). In this tool, the metasystem employs a transaction-oriented mechanism to modify the conceptual structures, so a transaction can be rolled back if the post-condition does not hold.

4.2 Evolution and the Model-View-Controller Design Pattern

In the traditional Model-View-Controller (MVC) design pattern [2], the model represents the data and behavior of the application domain, the view is some form of visualization of the state of the model, and the controller interprets the input of the user, modifying the model's state. In our case, since the metasystem is in charge of modifying the model, the controller must request the changes to the metasystem by sending the appropriate evolutionary actions. After evaluating and accepting the proposed actions, the metasystem will perform the modifications in the model, which, as usual in the MVC pattern, will notify its views so they get updated if necessary (Figure 10).

5. RELATED WORK

SEM-HP is grounded in the field of Adaptive Hypermedia Systems. In this field we can cite, among other relevant work, the AHA! architecture [5]. This architecture is based on the AHAM model [15], which, as SEM-HP, distinguishes a domain model, a user model and an adaptation model. Although they are conceptually similar in several

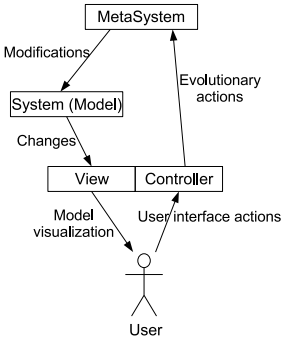


Figure 10: Evolutionary Model-View-Controller pattern

aspects, there are also significant differences. One important difference lies in the way the hypermedia is browsed: while in SEM-HP the user browses a conceptual structure, AHA! adopts a more conventional approach in which the user browses pages. In both systems there are rules that update the user model that are triggered by the user visiting elements in the hypermedia system, but AHA! allows to define any kind of update in the user model (knowledge, interest, etc.), while in SEM-HP update rules are used only to update the user knowledge, and interests and goals are defined in a different way. This may lead us to think that SEM-HP is more restrictive, but this is done with an idea in mind: evolution. A formal definition of, for example, update and knowledge rules has allowed to define algorithms that check the consistency of the hypermedia system while it evolves; in this case the metasystem checks that the knowledge and update rules will not lead to unreachable items (for example, due to unsolvable cycles in knowledge prerequisites). We consider that this evolutionary approach to adaptive hypermedia systems is the most remarkable contribution of SEM-HP.

Since this paper is focused in JSEM-HP, we will compare it to similar tools. There are several tools that are conceptually similar to JSEM-HP, specifically in relation to the usage of a knowledge representation model for semantically structuring and offering hypermedia information. These include Concilla [12] and CmapTools [3], both of which are still used and in further development.

Concilla is a "concept browser" which, as JSEM-HP, provides browsable maps with associations among concepts. Concilla has many differences from JSEM-HP, and we will not go into detail about it here. In Concilla, the browsable structures are called context maps, and, instead of the semantic networks used in SEM-HP it uses UML diagrams, so the user must learn UML notation in order to use them. Among other things, it focuses on the reutilization in different context maps of concepts and concept components (information), and, although they mention the reduction of user disorientation and knowledge overhead (web-surfing sickness), Concilla does not provide the user adaptation techniques present in JSEM-HP, as it does not provide its evolutionary support.

In this section we will focus on CmapTools, which is in some aspects very similar to JSEM-HP. CmapTools is a set of tools that allow building, navigating, sharing and criticizing knowledge models.

CmapTools is based on conceptual maps [13], which are used to represent knowledge by means of concepts and relations among them called propositions. Additionally, concepts can be enriched with resources, which can be links to pages, documents, images, etc. From this point of view, SEM-HP's conceptual structures can be seen as conceptual maps in which conceptual associations are used to form propositions and informational items correspond to CmapTools's resources.

An important difference between CmapTools's Conceptual maps and SEM-HP's conceptual structures lies in the representation of the items/resources associated to concepts. CmapTools lacks SEM-HP's functional associations, so it can not specify an item's role when associated to a concept (example, explanation, etc). Instead of this, resources are classified according to its type (image, web page, another conceptual map, pdf file, etc). We consider that the usage of functional associations is more convenient that CmapTools's approach, since we believe that for the user it is more important to know that an item contains, for example, an introduction to the concept, than knowing if this introduction is a web page or a pdf file (this is shown in SEM-HP through the item's attributes).

Continuing with knowledge representation, we would like to mention that CmapTools allows to nest concept maps within others, a feature that has been deliberately excluded in JSEM-HP because of additional disorientation problems it could cause to the user when navigating from one conceptual map to another. Complexity in conceptual structures is reduced by selecting a conceptual structure of presentation that is suitable to the user's features and goals, and by adapting it according to his knowledge.

On the one hand, CmapTools exceeds SEM-HP in the collaborative development and sharing of conceptual maps, for which includes a network of CmapTools servers in Internet. In addition, CmapTools has a wide user base and has been used in multiple environments while SEM-HP is starting to be used.

On the other hand, CmapTools lacks the evolution and user adaptation support present in JSEM-HP. Although CmapTools is based in well established knowledge representation models, it lacks the formal groundings that lie beneath JSEM-HP regarding the evolution of hypermedia systems (which is very important in SEM-HP), that allows not only the creation of consistent semantic hypermedia systems, but also their evolution so new changes do not cause inconsistencies in the system. In addition, SEM-HP is not only able to represent conceptual structures; it goes much further through the presentation, navigation and learning subsystems, adding advanced user adaptation features that allow the reduction of the cognitive overload and disorientation of the user browsing the conceptual structures.

6. CONCLUSIONS AND FURTHER WORK

SEM-HP is a model for the development of semantic adaptive hypermedia systems that takes into account the evolution of them. It groups several desirable features in a hypermedia system, both from the development and usage points of view. Firstly, its associated development process and evolutionary approach allow the evolution of the hypermedia system while maintaining the desired consistency. Secondly,

the semantic focus ensures that the information offered is given a semantic coherence, which is translated to the user via a navigable conceptual structure. Finally, the system can adapt to the user while he navigates, reducing the problems of disorientation and cognitive overhead, and making the user aware of his progress. We think that the main contribution of our approach is to provide semantics, adaptation and evolution together in a consistent framework.

The conceptual structures used in SEM-HP are very similar to conceptual maps, which have proved to be useful in education [4]. In addition, in SEM-HP the user's knowledge plays an important role in the user adaptation, which, besides the features mentioned, could in our opinion make the tool interesting for e-learning applications, in which the students can browse the offered material non-linearly and interactively, while being controlled by the update and knowledge rules and having their progress and knowledge gain monitored.

The initial development of the tool JSEM-HP overlapped with the late phases of the development of SEM-HP model, concretely the definition of the learning subsystem: knowledge and update rules, and user adaptation and guidance techniques such as guided routes [10]. At the beginning, the tool was used internally to check the feasibility of the most complex techniques, for example, that guided routes could be generated in a reasonable amount of time for real-world conceptual structures.

The tool aims to ease the development and maintenance of hypermedia systems by rejecting incorrect actions and keeping the system consistent by automatically propagating changes. Nevertheless, some elements of the tool have proven to be difficult to use for authors not familiarized with the model. Concretely, the editors for update and knowledge rules (figures 6 and 7) closely follow the formal representation in SEM-HP, so they are easy to use only if the author knows SEM-HP well. Now we are preparing the tool for its general usage, and this includes the development of a more intuitive user interface that will allow to define update and knowledge rules without having to use the current logical formula editor.

The future work includes the further development of the tool to include features of SEM-HP not mentioned in this paper, such as guided routes, other navigation modes, interesting items, feedback, etc. [8], and to use it in real world applications, in order to validate its usefulness and to correct possible problems. In addition, we are working in the JSEM-HP's interoperability, for which we are considering different graph and concept map interchange formate, such as GXL (Graph eXchange Language) [7] and XTM (XML Topic Maps) [11].

7. ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish MCYT's R+D project ADACO TIN2004-08000-C03-02.

8. REFERENCES

[1] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.

[2] Steve Burbeck. How to use model-view-controller (mvc). http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html, 1987.

[3] A. J. Caas, G. Hill, R. Carff, N. Suri, J. Lott, T. Eskridge, G. Gmez, M. Arroyo, and R. Carvajal. Cmaptools: A knowledge modeling and sharing environment. In J.D. Novak and F.M. Gonzlez, editors, *Concept Maps: Theory, Methodology, Technology. Proceedings of the First International Conference on Concept Mapping. Pamplona, Spain.* Universidad Pblica de Navarra: Pamplona, Spain, 2004.

[4] J. W. Coffey, M. J. Carnot, P. J. Feltovich, J. Feltovich, R. R. Hoffman, A. J. Caas, and J. D. Novak. A summary of literature pertaining to the use of concept mapping techniques and technologies for education and performance support, 2003. Technical Report submitted to the Chief of Naval Education and Training, Pensacola, FL.

[5] P. De Bra, A. Aerts, B. Berden, B. De Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *Proceedings of the ACM Hypertext Conference, Ottingham, UK*, 2003.

[6] Lina Garca-Cabrera, Mara Jos Rodrguez-Frtiz, and Jos Parets-Llorca. Evolving hypermedia systems: a layered software architecture. *Journal of Software Maintenance and Evolution: Research and Practice*, 14(5):389–405, 2002.

[7] Richard C. Holt, Andy Schrr, Susan Elliott Sim, and Andreas Winter. Gxl: A graph-based standard exchange format for reengineering, 2002.

[8] Nuria Medina-Medina. *Un modelo de adaptacin integral y evolutivo para sistemas hipermedia. El sistema de Aprendizaje de SEM-HP.* PhD thesis, Universidad de Granada, 2004.

[9] Nuria Medina-Medina, Fernando Molina-Ortiz, and Lina Garcia-Cabrera. Diversity of structures and adaptive methods on an evolutionary hypermedia system. *IEE Proceedings - Software*, 152(3):119–126, 2005.

[10] Nuria Medina-Medina, Fernando Molina-Ortiz, Lina García-Cabrera, and José Parets-Llorca. Personalized guided routes in an adaptive evolutionary hypermedia system. In *EUROCAST*, pages 196–207, 2003.

[11] Members of the TopicMaps.org Authoring Group. Xml topic maps (xtm) 1.0, http://www.topicmaps.org/xtm/index.html. 2001.

[12] Mikael Nilsson and Matthias Palmr. Conzilla - towards a concept browser. Technical report, NADA (Numerical Analysis and Computing Science), KTH (Royal Institute of Technology), Sweden, 1999.

[13] Joseph D. Novak. The theory underlying concept maps and how to construct them. http://cmap.coginst.uwf.edu/info/.

[14] Jos Parets, Ana Anaya, Mara J. Rodrguez, and Patricia Paderewski. A representation of software systems evolution based on the theory of the general system. *Lecture Notes in Computer Science*, 763:96–109, 1994.

[15] H. Wu, G. Houben, and P. De Bra. Aham: A reference model to support adaptive hypermedia authoring. In *Proceedings of the Conference on Information Science, Antwerp*, pages 51–76, 1998.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Systems Architecture 50 (2004) 407–416
JOURNAL OF SYSTEMS ARCHITECTURE
www.elsevier.com/locate/sysarc

A Software System evolutionary and adaptive framework: application to Agent-based systems ☆

Patricia Paderewski-Rodríguez [a], Juan Jesús Torres-Carbonell [b,*], María José Rodríguez-Fortiz [a], Nuria Medina-Medina [a], Fernando Molina-Ortiz [a]

[a] *Departamento de Lenguajes y Sistemas Informáticos. ETS Ingeniería Informática, Universidad de Granada C/Periodista Daniel Saucedo Aranda s/n, 18071, Granada, Spain*
[b] *Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información, Ministerio de Ciencia y Tecnología Palacio de Comunicaciones, Plaza Cibeles s/n, 28071, Madrid, Spain*

Available online 18 December 2003

Abstract

In this paper we present part of our current work: a proposal on a Software System evolutionary framework. This proposal is based mainly on previous work carried out by the GEDES (Group of Specification, Development and Evolution of Software) Research Group. Within this framework, we try to model the way a Software System can evolve, and especially, the evolution of Agent-based systems. We present the way systems evolve based on the application of operators and the understanding of definition of focusing on which should be these operators, and invariants in Agent-based systems, as well as introducing examples of actions and restrictions applied.

Keywords: Blackboard; Agent-based systems; Evolution; Software architecture

1. Introduction

We have in mind the existence of a Metasystem, introduced by Parets-Llorca [1] and applied in Rodríguez et al. [2,3], which is a system that belongs to the Software Development System and is isomorphic with the Software System. This Metasystem is necessary in order to perform modifications to the Software System, which affect its structure, or to produce new systems.

In order to go ahead with this project, we introduce a basic abstraction level composed by evolutionary mechanisms and evolutionary models [4,5]. These two concepts are related in such a way that the models are the way mechanisms are used. So, mechanisms are abstract evolutionary activities, that could or not could imply the modification of the structure of the system, and models are a representation of the way Software Systems evolve using those mechanisms.

☆ This research is supported by a project MEIGAS by the Spanish CICYT (TIC2000-1673-C06-04) which is a subproject of the DOLMEN project (TIC2000-1673-C06).
* Corresponding author. Tel.: +34-91-346-2764; fax: +34-91-346-1556.
E-mail addresses: patricia@ugr.es (P. Paderewski-Rodríguez), jj.torres@setsi.mcyt.es (J.J. Torres-Carbonell), mjfortiz@ugr.es (M.J. Rodríguez-Fortiz), nmedina@ugr.es (N. Medina-Medina), fmo@ugr.es (F. Molina-Ortiz).

1383-7621/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2003.08.012

The mechanisms proposed are

- *Adaptation Mechanism*: The way the system modifies its structure or function according to the necessities of the environment.
- *Inheritance Mechanism*: The way a new system is produced.

Models reflect the modifications of the Metasystem (the system used by the modeller to produce systems) and the system, as well as the inheritance of modifications acquired by these systems through their evolutionary record, by the use of the mechanisms. The models identified are the following:

- Metateleology directed by the modeller. It models the modification of the Metasystem by the modeller using the Mechanism of Adaptation.
- Teleology directed by the modeller. It models the modification of the system by the modeller using the Adaptation Mechanism.
- Inheritance of Acquired Metacharacters. It models the production of a new Metasystem that inherits the adaptations acquired by the old one, using the Inheritance Mechanism.
- Inheritance of Acquired Characters. It models the production of a new system that inherits the adaptations acquired by the old one, using the Inheritance Mechanism.
- Metasystem–Software System SelfAdaptation. It models the evolutionary adaptation of the Software System by interaction with the Metasystem and the application of the Adaptation Mechanism.
- Software System SelfAdaptation. It models the selfadaptation processes in which the Software System adapts to its environment without the direct intervention of the Metasystem, through the application of the Adaptation Mechanism.

As a part of our research, we suggest the correspondence between models of the evolutionary framework proposed, that follow a biological conceptual frame, and concepts of software engineering. This correspondence could be summarized in Table 1, in such a way that we could map both kind of concepts:

Table 1
Software engineering *versus* evolutionary framework

Software engineering	Proposed evolutionary framework
Genetic algorithms	Darwinian evolutionary paradigm
	Neolamarckian paradigm
No direct equivalence but code modification	Metateleology directed by the modeller
No direct equivalence but code modification	Teleology directed by the modeller
CASE Tools versioning	Inheritance of Acquired Metacharacters
System versioning and reuse	Inheritance of Acquired Characters
Automatic code generation using CASE tools	Metasystem-Software System SelfAdaptation
Learning systems	Software System SelfAdaptation

1. Darwininan evolutionary paradigm, based in natural selection, could be considered the basis of genetic programming.
2. Neolamarkism, closer to our proposal than Darwinian, is not so easily mapped. Nevertheless we can find the following correspondence between the models proposed and software engineering concepts:
   2.1. There is no explicit correspondence for the models named Metateleology directed by the modeller (that models the modification of the Metasystem by the modeller) and Teleology directed by the modeller (that models the modification of the system by the modeller) but the possibility of code modification.
   2.2. The model named Inheritance of Acquired Metacharacters could be mapped as CASE tools versioning.
   2.3. The model named Inheritance of Acquired Characters maps the development of different versions of a system and its reuse.
   2.4. The Metasystem–Software System SelfAdaptation model could be considered equivalent to automatic code generation using CASE tools.

2.5. Finally, Software System SelfAdaptation could map systems' learning and adaptive capacities.

2. The formalization: operators and conditions

The evolving models are formalized [5] applying some operators to carry out evolving changes in the systems, both the structural and functional modifications. Two kind of operators are differentiated in this formalization:

- *S-operators* to modify the structure. They will be used by the models with Adaptation Mechanisms by means of mutation–differentiation and Inheritance of Acquired Characters.
- *NS-operators* which do not modify the structure. They will be applied by the model that uses the mechanism of Adaptation by means of accommodation-learning.

These operators are applied taking into account the definition of several conditions over them and over their results.

Two kind of conditions are proposed:

- Conditions that should be true in the beginning of the system software life.
- System conditions to be satisfied in every situation. There are two particular cases of them:
  - Conditions that must be satisfied in concrete circumstances, not depending on time.
  - Conditions that have to be satisfied after a concrete sequence of occurrences.

This evolving framework is specified using this formalization based on operators and conditions.

We have presented a study [6] which specifies and applies the formalization to three kind of different systems: Agent-based systems [7,9], Hypermedia Systems [7,8] and Decision Support Systems [6].

In this paper we are interested in presenting with more detail how the evolving framework is applied to Agent-based systems. We will show how the operators and conditions model the evolving actions and restrictions of this kind of software systems.

3. The specification of systems based on agents

Following the features of our evolutionary and adaptive framework, the basic structure of an Agent-system will be proposed.

3.1. Characteristics of the evolutionary and adaptive framework

The elements of the evolving framework have to be specified to get running software systems. In order to instantiate the present case, the following aspects are going to be adopted.

The Metasystem is also a software system but its structure and functioning are always the same and never change. The modeller uses it as a CASE tool which allows the evolutionary activity to create and to modify the software system. This is represented in Fig. 1.

The evolutionary models used in the framework are

- *Inheritance of Acquired Characters.* Cloning is the mechanism used to create agents, which allows the reuse of existing agents. There is an elemental agent which has the basic components of every agent, which is used as a template for creating new agents from scratch.
- *Metasystem–Software System SelfAdaptation.* As Table 1 shows, the Metasystem is implemented as a CASE tool. The modeller interacts

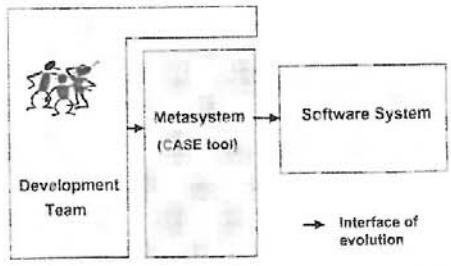


Fig. 1. Interaction between the development team, the Metasystem and the software system.

with the Metasystem to evolve the software system.

In addition, the operators are instantiated as functional and evolving actions. The functional actions correspond to the NS-operators because they do not modify the system structure. The system performs NS-operators. The evolving actions are S-operators and are carried out by the Metasystem to modify the structural elements of a system.

The conditions are expressed as restrictions or constraints, which insure that concrete system invariants hold, so the system integrity is guaranteed.

There are two kind of restrictions:

- Integrity restrictions: to assure the system integrity while functional actions are carried out.
- Integrity meta-restrictions: to guarantee the system integrity while evolving actions are carried out, it is, while it is evolving.

Following the work of Parets-Llorca [6], the correspondence between the concepts of the formal models and the concepts used in the *Agent-based systems* is presented in Table 2.

3.2. Structure of the Agent-based systems

The structure of the Agent-based systems and their components will be presented first in order to help the understanding of their evolution.

Table 2
Formalization in Agents-based systems

Formalization level	Agent-based systems
NS-operators	Functional action
Functional changes	Action execution
S-operators	Evolutionary action change propagation
Structural changes	Agent changes
Always conditions	Integrity meta-restrictions
Initial conditions	Initial integrity meta-restrictions
Static stage conditions	Integrity restrictions (action postconditions)
Temp. stage conditions	Integrity restrictions (pre, during conditions)

The active entities, which carry out the actions inside the system, are the *agents*. An agent [10] is autonomous, independent, and it is always prepared to work if its environment allow it. The agents of a system work concurrently. The agents that define a system are hierarchically related, therefore we used the terms of father-agent and son-agent. The son-agents carry out the actions requested by their parents. This relationship is called *aggregation* and allows the father-agents delegate to their son-agents. Another relationship is called *collaboration*, which permits agents with the same father (brother-agents) to cooperate in order to carry out more complex actions. Fig. 2 illustrates these relationships. In it, an aggregation relationship can be observed between *Agent 1.1* and *Agent 1.2* and their father, *Agent 1*. There is a collaboration relationship between *Agent 1.1* and *Agent 1.2* because both have the same father.

The actions carried out by the agents are *functional actions* and correspond to the NS-operators. The set of actions associated with the different agents in a system determines its functionality.

The actions have *attributes* to characterize them. There are two types of actions: *simple* or *complex*. The complex actions correspond to the definition of transactions. They are defined as a set of simple or complex actions carried out by different agents and in a concrete order. The system is not informed about the finalization of a complex action until all of the actions that are part of it have ended successfully (transaction atomicity). The complex actions of a system are related to the hierarchy of its agents, so each action in the description of a complex action of a father-agent should be carried out by a son-agent.

The actions have associated *pre-conditions*, *during-conditions* and *post-conditions*. They are the restrictions which influence the action activation taking into account the state of the system at each moment. A pre-condition must be verified before its associated action can be carried out. During-conditions can interrupt the execution of actions and post-conditions guarantee the integrity of the system after the action execution. Post-conditions imply carrying out the actions related to change propagation, if necessary.

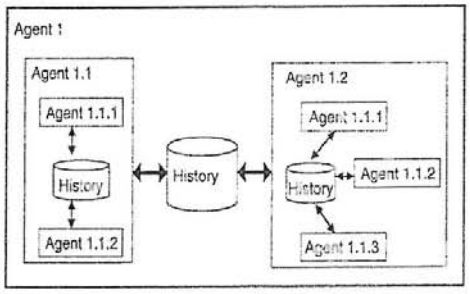


Fig. 2. Agent nesting.

There is a *history* to store the occurrences that have occurred in the system up to the present moment. The history represents the state of the system. It is also part of a blackboard architecture used to model the communication and coordination of the agents [11]. When an agent carries out an action, the information about the action (action occurrence) is stored in the history. When an agent needs to know if an action has been carried out, it queries the history. The history allows the aggregation and collaboration relationships.

There is a special agent called *Controller* associated to the history. It is in charge of managing the history and evaluating the restrictions associated to the actions, checking the information stored in it.

3.3. A special agent: the Controller

The structure described above presents three main problems:

1. The concurrent access for reading and writing in a central structure, the history.
2. The activation of the agents. In order to know when an action can be executed (and its agent activated), the agents have to repeatedly test the conditions of their actions, which implies a loss of efficiency.
3. The evaluation of conditions should be carried out without interference from new actions. When a condition is being tested, the history can be modified.

A special agent, called *Controller*, is introduced to solve these problems. This agent is in charge of providing services to the other agents, namely the evaluation of preconditions and the reception of new action occurrences. The *Controller* design follows the Precondition Dynamic Notifier (PDN) design pattern [9]. In addition, this agent guarantees the consistency of the history and provides an order to the agent requests. Each agent has its own *Controller* to manage its own history.

Fig. 3 shows the structure of the *Controller*, whose functionality is divided between the PDN and Evaluator agents. It also contains the history, which stores the occurrences of actions. The PDN agent stores occurrences of actions and activates agents. The Evaluator agent checks action preconditions and provides an answer to agents' queries.

The first problem, that of concurrent access, is solved by sequencing the read and write operations on the history in order to avoid possible inconsistencies.

The second problem stated above is especially important in the structural evolution of the system. This problem is solved by preventing an agent from testing the conditions when there are no possibilities of success. This implies that the Controller has knowledge about the agents, their actions and conditions, knowledge which changes dynamically as the system evolves.

The third problem is solved blocking the history while the agent Evaluator is checking a condition and establishing additional restrictions which provide mutual exclusion.

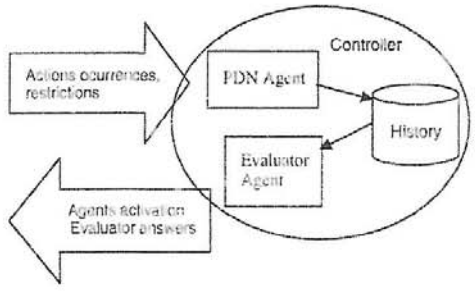


Fig. 3. Structure of the Controller agent.

3.4. The Transaction Manager

With the objective of guaranteeing the ACID (Atomicity Consistency Isolation Durability) properties of a transaction or complex action, a new special agent is designed: the Transaction Manager.

The structure of the Transaction Manager agent is similar to the Controller agent. Nevertheless, this is created when a transaction is activated and it disappears when the transaction finalises or is interrupted. The history it controls is a temporal history which stores the occurrences of the actions included in the transaction.

To define a transaction, its necessary use a specification language. We propose a language, called TDL (Transaction Definition Language) with the following operators:

- Sequence operator (;): $a ; b$ indicates that b is carried out when a has finished.
- Optional operator (|): $a \mid b$ express that a or b can be carried out. It is an exclusive or.
- Parallelism operator (||): $a \parallel b$ represents that a and b must be carried out. They can be carried out simultaneously.

The BNF description of the TDL language is the following:

```
<Transaction>
  ::=<Action><Operator><Action>|
  <Action><Operator><Transaction>|
  '('<Transaction>')'
<Action> ::= action_name
<Operator> ::= ';' | '|' | '||'
```

Several transactions can be executed concurrently. In this case, there is a Transaction Manager per each transaction. The Controller is in charge of creating the Transaction Managers when it receives stimuli and the system state is adequate.

When a transaction finalises successfully, the Transaction Manager sends the Controller the occurrence of this transaction. If the transaction is interrupted (because its during-condition does not hold), the Transaction Manager is eliminated and also its history without changing the system state.

4. Evolution of an Agent-based system

During the life-cycle of an Agent-based system, there are many circumstances which influence its changes. The running environment, the user skills or characteristics and the system requirements could change. These modifications imply the evolution of the system, that is, the addition of new agents, the elimination of others, the variation of the cooperation mechanisms,... A set of *S-operators* is created to allow the evolution of the different structural elements. The S-operators are the *evolving actions* that the Metasystem can carry out on the software system. They are

- Adding, deleting or nesting an agent in the agent hierarchy.
- Changing the definition of a complex action or transaction.
- Adding, deleting or modifying the restrictions of an action.

The full set of S-operators used to evolve Agent-based systems is shown in Table 3.

As we have shown, the Metasystem defines a list of *invariants* associated to the system, which always have to be satisfied (always conditions) or which have to be defined at the beginning of the functioning (initial conditions). The invariants depend on the system specification and on its desirable characteristics. The evolving actions have associated *meta-restrictions* responsible for guaranteeing these invariants.

As the Metasystem is also a software system, it has a *history* where it stores information about the occurrences of evolving actions. These are the changes carried out in the software system since it was created.

The Metasystem and the Software System are shown in Fig. 4. A development team is in charge of modifying the system using the Metasystem. The action interface is used by the final user to work with the system and by the development team to send stimuli to the Metasystem in order to change the system. The evolution interface allows the Metasystem to modify the Software System only if the meta-restrictions hold.

Table 3
S-operators or evolutionary actions in Agent-based systems

aAddSAction(A,P)	Adding a simple action *A* to an agent *P*
addCAction(A,P)	Adding a complex action *A* to an agent *P*
addPPred(X,A,P)	Associating or substituting a pre-condition *X* to an action *A* of an agent *P*
addTPred(X,A,B,P)	Associating or substituting a pre-condition *X* to a transactional action *A*, from the transaction *B*, of an agent *P*
addLAI(list,A,P)	Associating or substituting a list of incompatible actions, list, to an action *A* of an agent *P*
delAction(A,P)	Deleting an action *A* from an agent *P*
defCAction(def,A,P)	Modifying the definition *def* of a complex action *A* of an agent *P*
renameAction(A,B,P)	Renaming the action *A* from an agent *P* and calling it *B*
addAtt(att,A,P)	Adding an attribute *att* to an action *A* from an agent *P*
delAtt(att,A,P)	Deleting an attribute *att* from an action *A* from an agent *P*
renameAtt(att,att2,A,P)	Renaming an attribute *att* from an action *A* from an agent *P* and calling it *att2*
cloneAgent(P,Q)	Cloning the agent *P* and calling the new clon *Q*
createAgent(P,Q)	Creating an agent *P* as son of the agent *Q*
createSystem(P)	Creating an agent *P* without father, it will be the system
aggrAgent(P,Q)	Aggregating an agent *P* to another agent *Q*
disAgrAgent(P,Q)	Disaggregating an agent *P* from another agent *Q*
delAgent(P)	Deleting an agent *P*
renameAgent(P,Q)	Renaming an agent *P*, calling it *Q*
moveAction(A,P,Q)	Moving an action *A* from an agent *P* to another agent *Q*
moveAgent(P,Q,R)	Moving the agent *P* from *Q* to *R*

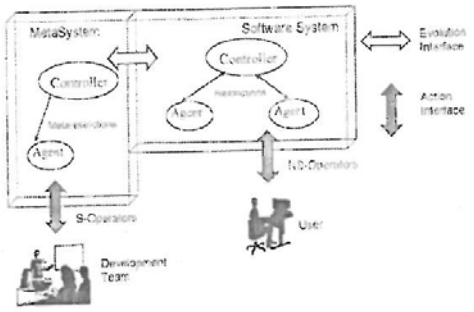


Fig. 4. Evolution of a Software System.

The meta-restrictions are verified using the system's history. They are expressed using a logic language [3] based on predicate temporal logic and are checked by the Controller.

This logic language consists of

- Vars (Variables): $x_1, x_2, y_1, y_2, \ldots$
- Const (Constants): $a, b, c, \ldots$
- Preds (Predicates) with the form $p(x^{-})$, where $x^{-} = \{x_1, x_2, \ldots, x_i\}$ represents the variables which can be unified or instantiated with values from the domain D for every kind of variable. Preds are operators and are shown in Table 3. $x^{-}, y^{-}, \ldots$ are used to distinguish different variable sets for different predicates.
- Logic operators: *and, or, not*
- Temporal operators: *since, once* ($\Diamond$)
- Quantifiers: $\forall$, $\exists$
- Formulas: a formula is an expression which is obtained recursively from the application of the operators on the predicates.
  Having $p(x^{-}), q(y^{-}) \in$ *Preds*:
  - $\Diamond p(x^{-})$: *not* $\Diamond p(x^{-})$ are formulas. These formulas are called *atomic formulas.*
  - The following expressions are formulas: $\Diamond p(x^{-})$ *and* $\Diamond q(y^{-})$; $\Diamond p(x^{-})$ *or* $\Diamond q(y^{-})$; $p(x^{-})$ *since* $q(y^{-})$; $\forall x_i \Diamond p(x^{-})$; $\exists x_i \Diamond p(x^{-})$ with $x_i \in x^{-}$. These formulas are called *simple formulas.*
  - If f is a simple formula and g is a simple or atomic formula, the following are also formulas: f *and* g; f *or* g; $\forall x_i\ f$; $\exists x_i\ f$; *not* f. These formulas are called *complex formulas.*

F models the conditions of the actions in an Software System.

4.1. Invariants

Eight invariants have been defined in the Agent-based systems:

1. *Unique names*: Every element in each hierarchical level of the software system have to be referenced without ambiguities.
2. *References*: Every element which exists in a software system may be referenced. For example, if the software system evolves and an element is deleted, its references should also be deleted.
3. *Attributes characterization*: Every simple or complex action must have at least one attribute.
4. *Possibility of carrying out actions*: The Metasystem should to guarantee that every action in a software system could be carried out in a moment, because its restrictions could be satisfied.
5. *Management of the agents aggregation graph*: the hierarchy of agents which defines a software system has to be consistent with the action complex definitions.
6. *Confluence*: The state of a system is determined by the actions carried out in each moment and by the order of them.
7. *Finalization*: The conditions of a system to finish or not finish should be specified. Many systems are characterized because they never finish but lock situations should be avoided.
8. *Unique actions*: An agent could have aggregated several son-agents, which could carry out actions with the same name and attributes. In this case, these actions should be the same because they work for the same father.

4.2. Examples of evolving actions and meta-restrictions

The evolving actions have associated meta-restrictions, which are their pre-conditions and post-conditions. The Metasystem verifies the pre-conditions and post-conditions respectively before and after carrying out an evolving action. It propagates the change, if necessary, and hence it guarantees that the invariants are verified.

Two examples of evolving actions will illustrate the use and the utility of the meta-restrictions or conditions.

Example 1. Evolving action to add a simple action, A_s, to an agent P:

$addSAction(A\_s, P)$

Its pre-condition states that the P agent should exist previously (preserving the reference invariant). Besides, no evolving *addSAction* with the same action name A_s is allowed in the Metasystem history (unique name invariant) since the modeller asked the Metasystem to carry out the evolving action, by sending an stimulus (*StimulusAddSAction*).

$$addSAction(A\_s, P) \leftarrow \Diamond exist(P) \text{ and } (not\ isInSSHA(A\_s, P)\ since\ StimulusAddSAction(A\_s, P))$$

The post-condition of this evolving action is associated with the invariant of the possibility of carrying out actions. It states that every action should to have associated a valid pre-condition previously (an action without pre-condition can never been carried out). The post-condition is expressed as

$$addPPrec(X, A\_s, P) \leftarrow \Diamond addSAction(A\_s, P)$$

Example 2. Evolving action to delete an agent P from the system:

$delAgent(P)$

Its pre-condition verifies the existence of the agent to be deleted (reference invariant). If the agent works for a father-agent carrying out concrete actions, those actions should also be associated with another son-agent. A stimulus is also necessary.

$$delAgent(P) \leftarrow \Diamond(exist(P) \text{ and } (\exists Q\ isFather(Q, P) \text{ and } \forall A\ isInSSHA(A, P)) \Diamond(\exists R\ isAgrAgent(R, Q) \text{ and } isInSSHA(A, R))) \text{ and } \Diamond StimulusDelAgent(P)$$

This evolving action has no post-condition but it is necessary to carry out some actions as part of the mechanism of change propagation. If an agent

is deleted, perhaps its sons should also have to be deleted. Therefore, the deletion is propagated through the whole branch of the deleted agent in the agents hierarchy.

5. Conclusions and future work

We have presented a software system evolutionary framework. A basic abstraction level composed by evolutionary mechanisms and evolutionary models has been presented. This framework was applied in defining the Hyperincursive and Anticipative characteristics of Software System Evolution [12].

In this paper, the mechanisms used are inheritance and adaptation. From the presented models, Inheritance of Acquired Characters and Metasystem-Software System SelfAdaptation are used.

This framework is applied to model an Agent-based systems dynamic architecture. In this architecture, we select the model of Inheritance of Acquired Characters using cloning.

The second model used allows the developer to use the Metasystem as a CASE tool to permit the evolution of the Agent-based system.

The Metasystem modifies the structure and functioning of the software system by a set of evolutionary actions, S-operators, allowing adaptations to the environment. These changes are allowed if the meta-restrictions associated with the evolutionary actions are verified, so the system invariants are preserved. Evolutionary actions allow additions to, deletions of or modifications to the components of the system (agents, actions and restrictions). A Controller agent maintains information about the system in the form of tables that are updated while the system evolves.

An agent can perform both simple and complex actions. For managing transactions (complex actions), a special agent (the Transaction Manager) is used, which preserves the properties traditionally associated to transactions (ACID).

This work is related to dynamic software architectures [13,14], in concrete to the ADL (Architecture Description Languages), which allow the design of the structure of a software system. But many ADLs are based on very complicated formalisms to be used in specifications of big software systems. The objective of some dynamic ADLs is the modification of the connections at run-time and do not consider that the components are active entities. Besides, some of them do not verify integrity restrictions or invariants before carrying out a modification in the software system.

In our model, any element or component of a software system could be modified and each change implies the verification of the integrity constraints associated to the evolving actions. The developer can construct the system in a gradual way: creating agents, associating actions to them, defining complex actions, and establishing the aggregation hierarchy and the model coordination/synchronization of the agents.

An interesting improvement is the application of the functionality of this architecture and of the coordination model in distributed systems, with the aim of adapting it to build systems based on mobile agents.

References

[1] J. Parets-Llorca, Reflexiones sobre el proceso de concepción de sistemas complejos, MEDES: un método de especificación, desarrollo y evolución de sistemas software, Doctoral Thesis, Universidad de Granada, 1995.

[2] M.J. Rodríguez, J. Parets-Llorca, P. Paderewski, A. Anaya, M.V. Hurtado, HEDES: A System Theory Based Tool to Support Evolutionary Software Systems, Lectures Notes in Computer Science, vol. 1798, Springer-Verlag, 2000.

[3] M.J. Rodríguez-Fortiz, J. Parets-Llorca, Using Predicate Temporal Logic and Coloured Petri Nets to specify integrity restrictions in the structural evolution, in: International Symposium on Principles of Software Evolution, IS-PSE2000, Kanezawa, Japan, November (2000), pp. 81–85.

[4] J.J. Torres-Carbonell, J. Parets-Llorca, Biological evolutive models applied to the evolution of software systems, in: E. Pessa, M.P. Penna (Eds.), Troisième Congrès Européen de Systémique, Rome, 1996, pp. 705–709.

[5] J.J. Torres-Carbonell, J. Parets-Llorca, A formalisation of evolution of software systems, in: F.R. Pichler, R. Moreno-Díaz, P. Kopacek (Eds.), Computer Aided System Theory—EUROCAST'99, LNCS 1798, Springer-Verlag, pp 429–449, 2000, ISSN 0302-9743.

[6] J. Parets-Llorca, A framework for software system evolution, in: INSA, International Conference in Honour of Herbert Simon, Lyon, 15–16 marzo, 2002.

[7] M.J. Rodríguez-Fortiz, P. Paderewski-Rodríguez, L. García-Cabrera, J. Parets-Llorca, Evolutionary modeling of software system: its application to agent-based and

Hypermedia Systems. in: International Workshop on Principles of Software Evolution, IWPSE 2001, Viena (Austria), Septiembre, 2001, pp. 54–61.

[8] L. García-Cabrera, M.J. Rodríguez-Fórtiz, J. Parets-Llorca. Evolving Hypermedia Systems: a layered software architecture. Journal of Software Maintenance and Evolution: Research and Practice 14 (2002) 389–405.

[9] P. Paderewski-Rodríguez, M.J. Rodríguez-Fórtiz, J. Parets-Llorca, An architecture for dynamic and evolving cooperative software agents, in: H.R. Arabnia, Y. Mun (Eds.), Proceedings of SERP'02 in WASA'02, Las Vegas, USA, Junio 2002, pp. 44–50.

[10] S. Franklin, A. Graesser, Is it an Agent, or just a Program? A taxonomy for autonomous agents, in: Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, Berlin, 1996.

[11] D. Gelernter, N. Carreiro, Coordination languages and their significance, Communications of ACM 35 (2) (1992).

[12] J.J. Torres-Carbonell, J. Parets-Llorca, D.M. Dubois, Software Systems evolution, free will and hyperincursivity, in: D.M. Dubois (Ed.), International Journal of Computing Anticipatory Systems, CHAOS, Liège, Belgium, vol. 12, Soft Computing and Computational Intelligence: Cognitive, Neural, and Psychoanalytical Anticipation; Computer Science, Data and Knowledge Systems, Partial Proceedings of the Fifth International Conference CASYS'01 on Computing Anticipatory Systems, 17–18 August, 2001, Liège, Belgium, 2002, pp. 3–24.

[13] R. Torkar, Dynamic software architectures, Extended Report for I. Crnkovic, M. Larsson (Eds.), Building Reliable Component-Based Systems, Artech House, July 2002. ISBN 1-58053-327-2: Architecting Component-Based Systems, J. Bosch, J.A. Stafford (Chapter 3).

[14] J. Andersson, Issues in dynamic software architectures, in: Proceedings of the Fourth International Software Architecture Workshop (ISAW'4), ICSE'2000, 2000.

Dr. Patricia Paderewski-Rodríguez is an Associate Professor in the Departamento de Lenguajes y Sistemas Informáticos at the University of Granada (Spain). Her research publications are in the areas of software systems evolution models, cooperative Agent-based systems, object oriented methods and operative systems. She received her Ph.D. from the University of Granada.

Dr. Juan Jesús Torres-Carbonell is a part time Professor in the Departamento de Informática at the University Carlos III in Madrid (Spain) and currently is working at the Spanish Ministry of Science and Technology. His research publications are in the areas of software systems evolution models, anticipatory systems, object oriented methods. He received his Ph.D. from the University of Granada.

Dr. María José Rodríguez-Fórtiz is a Full Professor in the Departamento de Lenguajes y Sistemas Informáticos at the University of Granada (Spain). She is Research Director of the group of Software Development and Evolution at the same department. Her research publications are in the areas of software systems evolution models, formal methods for specification, object-oriented methods, tools and languages. She received her Ph.D. from the University of Granada. She is a reviewer of Information and Management Journal (North-Holland).

Mrs. Nuria Medina-Medina is an Associate Professor in the Departamento de Lenguajes y Sistemas Informáticos at the University of Granada (Spain). She researches and have publications in the areas of software systems evolution models and adaptive hypermedia models.

Mr. Fernando Molina-Ortiz is an Associate Professor in the Departamento de Lenguajes y Sistemas Informáticos at the University of Granada (Spain). He has been working in a research project implementing a tool related to software systems evolution models and adaptive hypermedia models. His publications are in these areas.