INTEGRACIÓN DE LOS REQUERIMIENTOS DE CONCIENCIA SITUACIONAL Y GRUPAL AL DESARROLLO DE SISTEMAS COLABORATIVOS Y DINÁMICOS USANDO UN ENFOQUE BASADO EN MODELOS

por

José Figueroa Martínez

Una Memoria de Tesis Doctoral Presentada como
Requerimiento para optar al Grado de
Doctor por la Universidad de Granada vinculado al Posgrado en Desarrollo de Sistemas Software

Directores:

Francisco Luis Gutiérrez Vela, Universidad de Granada, Granada, España

Víctor Manuel López Jaquero, Universidad de Castilla-La Mancha, Albacete, España

UNIVERSIDAD DE GRANADA

Departamento de Lenguajes y Sistemas Informáticos

Septiembre 2012

Editor: Editorial de la Universidad de Granada
Autor: José Figueroa Martínez
D.L.: GR 358-2013
ISBN: 978-84-9028-341-7

Muchas personas me han apoyado durante este periodo doctoral al lado de mi esposa Lluvia en España. Ha sido una experiencia hermosa e inolvidable para nosotros y aunque hemos pasado momentos muy difíciles nos llevamos mucha alegría y amistades de España. Doy un agradecimiento muy especial a las siguientes personas:

* A mi esposa Lluvia, por su confinaza, apoyo, cariño y su creciente paciencia en tiempos difíciles.

* A mi padre (en paz descanse) y a mi madre quienes me han guiado y apoyado para salir adelanteante a pesar de la distancia y de las dificultades.

* A mis hermanos Reyes, Haydee, Leonardo y Francisco. De todos ustedes tengo un poco y me alegra mucho que sean mis hermanos.

* A mis cuñadas(os) Martha, Aimé, Rubén y Karla y a mis hermosos sobrimos Daniel, Karen, Jimena, Leonardito, Ricardo y Eduardo.

* A mis suegros Rosa Irene y Miguel Arturo y a mi cuñada Nayeli por ser mi segunda familia en Hermosillo.

* A mi amigo Eduardo (sacerdote) que me apoyo en momentos muy difíciles estando lejos de mi familia.

* A los amigos que estuvieron cerca pasando en momentos alegres y no tan alegres. Norman, un abrazo.

* A los amigos que no estuvieron tan cerca pero estuvieron ah: Odette, Carlos, Eliseo, un abrazo también.

* A Clara y a su hermosa familia (Jenaro, Paqui, Miriam y Esperanza). Han sido para Lluvia y para mí nuestra familia en España. Gracias por todo. Abrazo cariñoso para José y Manoli, Miriam.

* A mis hermanas (espiritualmente hablando) las Agustinas Recoletas de Granada: Sandra (mi hermana de corazón), María Clara, Paulina, Miroslava, Erlinda, Karem, Paquita, Teresa, Genoveba, Gloria, Conchis, Feliza, Lucresia y la madre Nieves. Son un amor de mujeres. Gracias por hacernos parte de su familia.

* A mis hermanas (también espiritualmente hablando) las Misioneras Combonianas en Granada: Ida, Cidalia, Silvia, Palmira, Loretta, Ana, Yousi y Chiara. Gracias por compartir con nosotros trabajo, comidas y fiestas inolvidables. Son maravillosas.

* A mis amigos religiosos el padre Paco y José Gabriel de la parroquia de la chana.

* A los miembros del proyecto Girasol como Asención, Jerónimo (con su esposa Ana y sus hijos), Andrés, Ana, Paco, Peregrino, Ángel. Mis mejores deseos para ustedes y para el proyecto.

* A mis alumnos del proyecto Girasol como Maodo, Abú, Izaca (hermano de Abu), Tierno, Moussa (los dos), Loretta, Elena y los demás de los que no me acuerdo su nombre pero estuvieron ahí conmigo compartiendo su tiempo.

* A mis compañeros del Aikido, les agradezco cada clase que compartí con ustedes y cada golpe y derribe que nos dimos. Gracias Jose (sensei y buen amigo), Karina y Gabriel (esposa e hijo del sensei), Jose electricista y Jose que tiene su caballo, Rosa, María, Carlos, Maxi, Tai, Daniel, Andrés y todos los que conocí y aprecié practicando este hermoso arte.

* A mi amiga Ana con la que traté de aprender un poco del arte de las castañuelas. Gracias por tus enseñanzas y por compartir tu arte.

* A mis amigos durande el periodo de tesis. Pasé muy buenos momentos con ustedes. Hernán, Javier, Sandra, Álvaro, Sergi, Rafa, Carlos, Úrsula, Natalia, varios Joses. Un fuerte abrazo.

* A mis amigos cortijeros, con los que compartí buena comida y buena convivenvia. Miriam, Esperanza, Inma, Luis, Jose, Jesús (varios), Sebastian, Juan, Isabel, Javier y la pequeña Miriam.

* A mi asesor Francis y a Patricia, su esposa. Solo he recibido apoyo por su parte y en verdad se los agradezco.

* A mi familia poblana nos reciben siempre con los brazos abiertos: Tía Maricela, tía Ciria, tía Ortencia, tío Augurio, tía Virginia y tía Ines (esposa de

Arón). A mis primos Amelia, Javier, Edith, Kaita, Lida, Tere, Aurora, Jero, Danitza, Lupita, Emanuel, Juan, y la cantidad grandísima de sobrinos y sobrinas preciosas que tengo por esas tierras.

* Les deseo lo mejor familia y amigos míos que me acompañaron de alguna forma durante la duración de esta aventura fuera de mi madre patria, México.

AGRADECIMIENTOS

Quiero agradecer a todos los que me guiaron o hicieron sugerencias durante este proceso, pero especialmente a las siguientes personas:

- Francisco Luis Gutiérrez Vela, mi asesor en la Universidad de Granada desde el principio de la investigación, por su amistad, su apoyo y por su guía en este periodo de investigación.

- Víctor López Jaquero, mi asesor de las tierras manchegas de Albacete, por su amistad y apoyo en los aspectos más complicados de la investigación. Gracias a tí y a Francis por ayudarme a terminar este proceso de maduración personal y profesional.

- Pascual González por sus ideas y su soporte conjunto con Víctor. Mi más sincera admiración para ustedes y su trabajo.

- Cesar Collazos, por sus excelentes ideas y por su ejemplo de amor a la investigación. Gracias por tu apoyo.

- Luis Castillo Vidal, Juan Fernández Olivares, Antonio González Muñoz, José Manuel Benítez Sánchez y Raúl Pérez Rodríguez, miembros investigadores del proyecto Integra, por sus enseñanzas y por su cordial trato durante el proyecto. He aprendido muchas cosas buenas de cada uno y les agradezco el tiempo compartido con ustedes durante la duración del proyecto.

- A los profesores del Departamento de Lenguajes y Sistemas Informáticos, por guiarme a través del mundo de la investigación durante mi periodo de Maestría en Desarrollo de Software.

- Y finalmente, a mis profesores que a lo largo de mi educación lograron enseñarme los conocimientos necesarios para tener una vida productiva y

ética. Gracias a los profesores Pedro Flores, Lupita Cota, Irene Rodríguez, Donald, entre otros profesores que marcaron mi vida de alguna manera.

ÍNDICE GENERAL

Índice de Tablas

Índice de Figuras

CAPITULO 1

Introducción

El desarrollo de software es el proceso de construcción y mantenimieto de las aplicaciones y sistemas software que se utilizan hoy en día para todo tipo de aplicaciones. Desde los servicios en la Web para comprar, vender o estar comunicados con familiares y amigos hasta el control de nuestros aparatos electrodomésticos. Cada sistema o aplicación software tuvo un proceso de construcción acorde a las necesidades y posibles usos de dicho software.

El campo de interés de este trabajo son los sistemas colaborativos o "*Groupware*[1]". Dichos sistemas se identifican por tener un uso muy particular: el dar soporte al trabajo en grupo, ya sea parcial o íntegramente a través del ordenador.

En un sistema Groupware un usuario puede realizar actividades en las que intervienen otros usuarios, de tal forma que puedan combinar sus capacidades y trabajo para conseguir un determinado objetivo.

Muchos tipos de sistemas entran en esta categoría aunque han tomado nombre propio dada su utilización, como por ejemplo, la redes sociales. La actividad realizada en una red social depende de los usuarios, y puede ser laboral o lúdica en su totalidad. Tal vez el mayor uso de una red social es estrechar las relaciones entre personas que no pueden comunicarse cara a cara. Esta característica aunada a la facilidad de acceso a través de Internet ha hecho que las redes sociales sean ampliamente utilizadas en todo el mundo.

[1] `http://en.wikipedia.org/wiki/Groupware`

Al integrar el soporte del trabajo grupal, los sistemas Groupware adquieren un conjunto de dificultades que se ven reflejadas en su proceso de desarrollo. Una de estas dificultades es un aspecto esquivo y muy abstracto llamado "*conciencia*" o en inglés "*awareness*" (Awareness). Pero, ¿Qué es la *conciencia* y qué tiene que ver con el software?

Como veremos en este trabajo, la *conciencia* o Awareness tiene muchas interpretaciones y dependiendo de ellas, un determinado grado de inclusión en el software o por lo menos de manera directa.

Especialmente en el Groupware, el Awareness es manejado como la "*conciencia grupal*". En otro tipo de sistemas es manejado como "*conciencia situacional*". Para manejar un solo término lo suficientemente descriptivo, manejaremos el término "Awareness" para referirnos a los dos conceptos previos que están más relacionados de lo que parece, y que como veremos más adelante, se engloban en lo que es el Awareness.

Un aspecto importante de este trabajo es que se enfoca en el desarrollo dirigido por modelos [Hailpern and Tarr, 2006]. La razón para ello es que los sistemas Groupware tienen más requerimientos de información que otro sistema que no soporte el trabajo grupal. Los procesos interactivos entre humanos soportados por computadora han demostrado ser complejos de soportar y desarrollar [Gutwin and Greenberg, 2004].

El desarrollo dirigido por modelos busca mantener una estructura de desarrollo trazable desde los modelos conceptuales del sistema hasta su representación final en las interfaces de usuario finales. Nuestro interés es poder aprovechar este enfoque para encontrar el punto en el que se conectan las interfaces de usuario para proporcionar Awareness y el Awareness representado y soportado por el sistema.

Buscamos mejorar los sistemas Groupware integrando el Awareness al proceso de desarrollo siguiendo un enfoque diridigo por modelos, de tal forma que a través

de la representación del Awareness a nivel de requerimiento podamos propagar su influencia a lo largo del proceso del desarrollo, manteniendo así la máxima de la trazabilidad de requerimientos y mecanismos de transmisión de información.

1. Preliminares

Este trabajo se desarrolla en el marco del proyecto del Ministerio de Ciencia e Innovación de España, Calidad, Adaptación y Nuevos Paradigmas Aplicados a Sistemas Colaborativos, subproyecto ADACO (Sistemas Adaptativos y Colaborativos con Soporte Web) TIN2008-06596.

Parte de los objetivos del proyecto estaban relacionados con mejorar el desarrollo de los sistemas colaborativos o Groupware usando un enfoque dirigido por modelos (*Model driven development* o MDD), en el cual el sistema es modelado parcial o totalmente y la aplicación es generada a partir de dicha especificación por sucesivas transformaciones entre modelos.

En nuestra investigación sobre los sistemas colaborativos, su estructura y sus componentes encontramos varias referencias a un aspecto relacionado con el soporte de la colaboración implícito en los sistemas Groupware: **la conciencia de grupo**.

Varios trabajos en la literatura sobre el Groupware apuntaron a el concepto de "*Awareness*" definido por Paul Dourish y descrito como el conocimiento que se tiene de las actividades de los otros miembros durante el trabajo en equipo [Dourish and Bellotti, 1992]. De esta forma el concepto del Awareness para una parte de la comunidad científica está relacionado directamente con el trabajo en equipo por definición.

Ya que el Awareness parecía ser un aspecto tan importante comenzamos la investigación para determinar si era necesario incluirlo en el proceso de desarrollo de los sistemas Groupware usando el enfoque MDD.

Desde el principio de la investigación nos percatamos de que el Awareness no

es un concepto fácil de entender dada sus múltiples interpretaciones y su vaga definición. Este y otros problemas han sido descritos por [Schmidt, 2002] y en el caso del desarrollo de software un problema muy importante es la falta de un modelo conceptual, lo cual es normal si tomamos en cuenta que hay más de una definición aceptada del Awareness en lo que respecta al mundo del software.

Para entender más al Awareness decidimos abrir el rángo de investigación y cubrir un poco más de terreno fuera de la lituratura científica referente al desarrollo de software y al Groupware. Ahí es donde encontramos las bases teóricas para nuestro trabajo [Endsley, 1995, Gutwin and Greenberg, 2004, Carroll et al., 2006, Drury and Williams, 2002].

Varias preguntas debían responderse para cumplir nuestros objetivos básicos. La principal pregunta es **¿Qué es el Awareness?**.

Otras cuestiones también deben responderse, tales como su uso, su importancia, su estructura, las dificultades que conlleva, etc. Dichas cuestiones nos permitirán desarrollar los herramientas y técnicas adecuadas para lograr nuestros propósitos.

Encontramos que la relación usuario →retroalimentación se ha mantenido como base para el soporte de la interacción. En este trabajo le damos otra visión a la necesidad de la retroalimentación por parte de los usuarios.

La conciencia del entorno o Awareness representa el *¿por qué?* de la necesidad de proporcionar retroalimentación o "*feedback*" (Figura 1) a los usuarios de un sistema interactivo.

En base a esta premisa (hipótesis tal vez), creemos que mejorando el soporte del Awareness podemos mejorar drásticamente el soporte de la interacción persona-ordenador fuertemente requerida en sistemas interactivos y especialmente en el caso de los sistemas Groupware.

Nuestro trabajo se centra en confirmar la importancia del Awareness en el Groupware (sin descartar los sistemas no colaborativos) y en integrar el Aware-

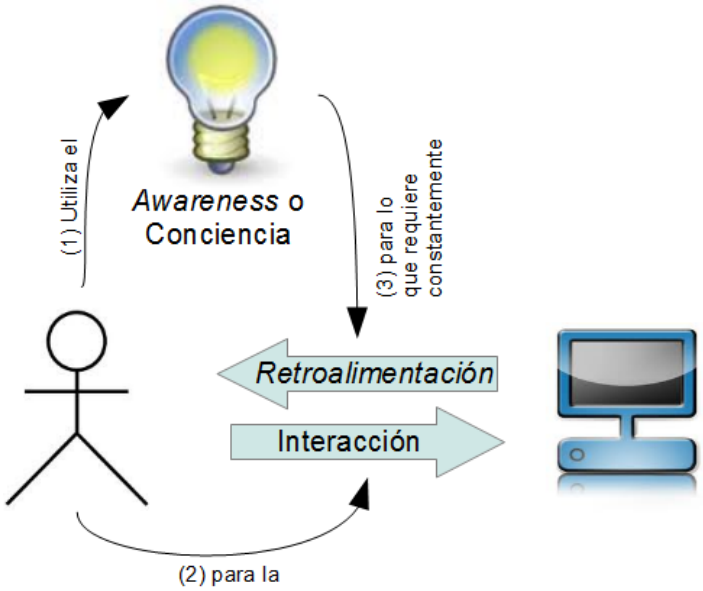


Figura 1. Relación entre el usuario, el Awareness y la retroalimentación proporcionada por el sistema software.

ness como parte del proceso de desarrollo, especialmente en el desarrollo dirigido por modelos.

2. Objetivos

El objetivo principal de la tesis es mejorar el soporte de la colaboración e interacción de los sistemas interactivos a través del desarrollo de los modelos necesarios para incluir el soporte del Awareness en el proceso de desarrollo de sistemas usando un enfoque dirigido por modelos.

Para lograr dicho objetivo desarrollamos las siguientes actividades detalladas a lo largo de la tesis:

- Una descripción general de los sistemas Groupware, tanto en sus componentes generales como en la descripción de algunas metodologías de desarrollo específicas.

- Revisión del estado del arte sobre el proceso de Awareness en general y su relación con los sistemas colaborativos.
- Catalogación de diversas formas de interacción especialmente orientadas a proporcionar Awareness.
- Definición de un modelo conceptual amplio sobre el Awareness.
- Definición de una forma de caracterizar el Awareness en un dominio de trabajo.
- Definición de un modelo para especificar requerimientos de Awareness.
- Integración del soporte del Awareness a una metodología de desarrollo a nivel de requisitos.
- Definición de las bases conceptuales y técnicas para unificar el soporte del Awareness durante el desarrollo del software.

3. Estructura de la tesis

Comenzamos este documento de tesis con la Introducción (Capítulo 1), el cual describe nuestro propósito, las razones por las cuales decidimos realizar este trabajo y el entorno en el cual lo desarrollamos.

Posteriormente presentamos un estudio general de los sistemas Groupware (Capítulo 2) centrándonos en algunas metodologías de desarrollo específicas para este tipo de sistemas. Estas metodologías nos permiten conocer de manera general el nivel al cual soportan la llamada conciencia situacional y grupal, a fin de ver las necesidades que no están cubiertas en dichas metodologías y en otras a nivel general.

Después presentamos un estudio profundo del concepto de conciencia situacional y grupal o "*Awareness*" (Capítulo 3) aplicado al software interactivo y principalmente al Groupware. Dicho estudio nos permitirá comparar puntos de

vista, técnicas de soporte y otros aspectos que influyen en el manejo de este concepto en el desarrollo del software, así como su importancia en todo el proceso de trabajo interactivo, colaborativo y cooperativo.

El estado del arte sobre el Awareness nos permite también conocer más a detalle su composición, su propósito y las dificultades que conlleva su soporte en el software. De esta forma podemos delimitar nuestra propuesta y justificarla en base a lo encontrado en este capítulo.

El Capítulo 4 presentamos la propuesta para modelar conceptualmente el Awareness, así como un modelo para requerimientos de Awareness y el marco de referencia sobre el cual se basa el modelo de requerimiento.

Posteriormente introducimos la extensión para el soporte del Awareness a nivel de requirimientos (Capítulo 5) en metodología y lenguaje de desarrollo de sistemas interactivos.

Para finalizar concluimos este trabajo con las conclusiones generales (Capítulo 6), las cuales incluyen nuestras aportaciones y la descripción de algunos temas que quedan abiertos como parte del trabajo futuro.

CAPITULO 2

Groupware

1. Introducción

Los sistemas Groupware son una evolución natural de los sistema mono usuario. Es decir, el ser humano ha creado entornos virtuales para realizar sus actividades. Si varias de las actividades del hombre son colaborativas o cooperativas, es lógico pensar que en algún momento se requerirían entornos virtuales para soportar dichas actividades.

El trabajo colaborativo soportado por ordenador comenzó a niveles básicos desde que se construyeron las primeras computadoras. En el tiempo en el que los ordenadores eran enormes y leían tarjetas, varias personas podían utilizarlos para apoyarse en tareas grupales, aunque fuera durante un limitado espacio de tiempo. Cada usuario podía aportar a la tarea grupal con el resultado de sus cálculos en el ordenador compartido, ayudando a completar la tarea en general.

Sin duda alguna, el primer entorno virtual de trabajo compartido fue la línea de comandos de los sistemas UNIX [Ritchie and Thompson, 1974]. En este punto, más de un usuario podía trabajar en el sistema operativo sirviéndose de comandos y aplicaciones básicas. Los mecanismos ahora reconocidos como necesarios para el trabajo colaborativo ya estaban presentes en estos sistemas a través de comandos[1]:

[1] `http://man.cat-v.org/unix-1st/`

- **who**[2]: Lista los usuarios conectados al sistema.
- **write**[3]: Permite mandar un mensaje a un usuario en el sistema.

Estos dos comandos permiten saber quién está disponible en el sistema y permiten mandar mensajes para mantener una comunicación escrita. Esta información es básica para cualquier sistema colaborativo actual que se desarrolle sobre un espacio compartido [Gutwin and Greenberg, 2002].

Llamamos entorno virtual a cualquier entorno soportado por el ordenador, ya sea a través de interfaces gráficas, auditivas, táctiles, en línea de comandos, etc. Cualquier conjunto de interfaces de usuario utilizadas para interactuar con algún sistema conforman un entorno virtual.

La diferencia entre estos primeros sistemas informáticos y lo que existe en la actualidad es abismal.

Hoy en día, las "nuevas tecnologías" ya no son tan nuevas. Las personas con acceso a Internet se comunican a través de redes sociales, compran productos en base a recomendaciones de otros usuarios, se divierten con juegos en línea o trabajan a través de sistemas que no están hospedados en sus ordenadores, sino en servidores externos o en Internet.

Varios procesos de nuestra vida diaria ya no se realizan en el mundo real, sino en los entornos virtuales soportados por el ordenador. Estamos en la era digital.

Ciertamente no todos en el mundo tienen acceso a Internet, a las "nuevas tecnologías" o a un ordenador personal. En el caso de comunidades empobrecidas sin recursos para montar redes de ordenadores o para conectarse a Internet, es normal que los ordenadores se compartan entre los miembros de la comunidad. Aún en estos entornos tan limitados, el uso de sistemas Groupware es factible a través de trabajo asíncrono hecho por diferentes personas en el mismo ordenador.

El término "*Groupware*" fue usado para designar al software colaborativo a

[2] `http://man.cat-v.org/unix-1st/1/who`
[3] `http://man.cat-v.org/unix-1st/1/write`

finales de los años 1980, cuando [Richman and Slovak, 1987] escribieron:

"Como un tendón electrónico que une a los equipos de trabajo, el nuevo *groupware* busca situar a los ordenadores en medio de las comunicaciones entre gerentes, técnicos y cualquier otro que interactue en grupos, revolucionando la forma en la que ellos trabajan."

En dicho escrito se muestra implícitamente el objetivo de los sistemas Groupware: mejorar el trabajo en equipo usando el ordenador como intermediario para ello.

La importancia de los sistemas Groupware radica en **su capacidad de ofrecer un entorno virtual para trabajar en grupos**, permitiendo a los usuarios realizar sus actividades desde cualquier parte de la red de su oficina o desde cualquier parte del mundo vía Internet, ya sea en tiempo real o de forma asíncrona. Dicha capacidad u objetivo es también una de las razones por las cuales desarrollar sistemas Groupware es complejo: el soporte del trabajo en grupo.

Como veremos en la Sección 2 los sistemas Groupware se han especializado en distintos tipos dependiendo de su propósito: Trabajo en grupo (*Computer Supported Cooperative Work*), aprendizaje en grupo (Computer Supported Collaborative Learning), investigación en grupo (Computer Supported Collaborative Research), entre otros.

No podemos olvidar la rama lúdica o semi lúdica de los sistemas Groupware: Los juegos y las redes sociales, los cuales son tal vez los sistemas informáticos de trabajo u ocio en grupo más usados del mundo.

La necesidad de soportar el trabajo en grupo y todas sus dificultades ha generado el desarrollo de metodologías de desarrollo específicas para los sistemas Groupware, de las cuales analizamos algunas de ellas en la Sección 3.

Como veremos en el Capítulo 3, la necesidad de información actualizada y propicia para permitir la interacción entre los usuarios, el ordenador y los demás usuarios se incrementa en los sistemas Groupware.

No todos los componentes del Groupware 4 están relacionados con el soporte de la colaboración. Sin embargo, a lo largo de este capítulo constataremos que el soporte de la interacción y del trabajo en grupo repercute en todo el sistema y por lo tanto, en gran parte del proceso de desarrollo.

2. Tipos de sistemas Groupware

El trabajo grupal se refiere a las actividades que se realizan en grupo, ya sea entre humanos o agentes artificiales.

No obstante, los tipos de sistemas Groupware más notorios son los siguientes:

- Sistemas para el trabajo colaborativo o CSCW.
- Sistemas para el aprendizaje colaborativo o CSCL.
- Sistemas para la investigación colabortiva o CSCR.
- Juegos.
- Redes sociales.

El trabajo de [Hinze-Hoare, 2006] distingue al CSCW por tener un espacio de trabajo, al CSCL por tener un espacio de aprendizaje y al CSCR por tener un espacio de investigación.

Un sistema Groupware puede pertener a varias de las categorías anteriores y poseer varios espacios para trabajo, aprendizaje, investigación u ocio. Por ejemplo, un juego puede ser utilizado para el aprendizaje colaborativo y como medio de capacitación para un trabajo grupal fuera del ordenador.

A continuación describimos las peculiaridades de cada uno de estos tipos de sistemas Groupware.

2.1. Sistemas CSCW

Las siglas CSCW representa *Computer Supported Collaborative Work* o Trabajo Colaborativo Soportado por Computadora. En este grupo entran los sistemas diseñados para soportar el *trabajo* en grupo, ya sea a través de tareas cooperativas o colaborativas.

Entendemos una tarea colaborativa como una tarea grupal en la que se trabaja en un espacio de trabajo compartido o utilizando artefactos u objetos compartidos por todos los miembros del equipo. Entendemos una tarea cooperativa como una tarea grupal en la que se dividen las actividades a realizar entre todos los miembros del grupo, dando lugar a varias tareas individuales o igualmente colabortivas/cooperativas [Dillenbourg et al., 1996].

Los CSCW se centran en sistemas para el trabajo. Es decir, sistemas administrativos, de manejo de proyectos, apoyo a toma de decisiones, monitorización y control de procesos, entre otros. La idea general es que con el trabajo en grupo se obtengan productos o se completen tareas que beneficien a los miembros del grupo como conjunto. Siendo más específicos, estos productos o tareas pueden estar relacionadas con el aprendizaje o con el ocio. De ahí que los sistemas CSCL sean considerados un subconjunto de los sistemas CSCW [Hinze-Hoare, 2006].

Como ejemplos de sistemas CSCW encontramos los editores colaborativos como GoogleDocs[4] u otros más especializados como Etherpad[5], cuyo objetivo es ser un editor de texto colaborativo en-línea *open source* para uso general (Figura 2). Dicho editor muestra de forma sencilla (colores) los espacios en los cuales está trabajando cada usuario.

Para la gestión de proyectos y grupos encontramos el Groupware BSCW[6] (*Basic Support for Cooperative Work*), el cual, además de no estár muy especializado para ser de propósito general, también ha sido utilizado para experimentar

[4] `http://docs.google.com`
[5] `https://github.com/Pita/etherpad-lite`
[6] `http://www.bscw.de`

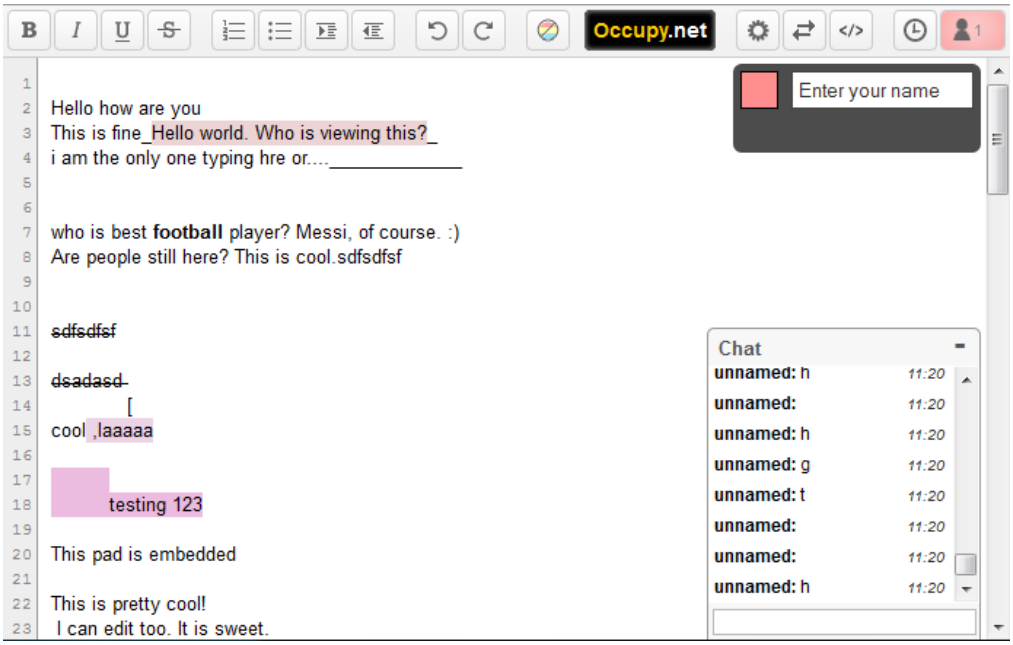


Figura 2. En el espacio de trabajo compartido de EtherPad Lite pueden verse herramientas comunes para la comunicación con el *chat* o la visualización de los usuarios conectados.

con simulaciones gráficas [Ahmed and Brahim, 2008].

Un tarea grupal muy importante es la toma de decisiones en grupo. Para este propósito existen los sistemas de apoyo a la toma de decisiones (DSS) grupales, los cuales son sistemas interactivos basados en computadora que facilitan la solución de problemas por los usuarios que toman las decisiones trabajando como un grupo. Este tipo de sistemas Groupware incluyen comunicación, colaboración y soporte a la toma de decisiones como principal objetivo en un contexto y dominio específico, aportando los medios necesarios para la comunicación y colaboración, y para la toma de decisiones al proveer la información necesaria para ello [Power and Sharda, 2009].

Si la realización en grupo de una tarea no es el objetivo principal, tal vez lo es el propio proceso de colaboración en el cual se crean relaciones sociales y se adquiere conocimiento de los otros miembros. En este caso los sistemas Groupware se especializan en los sistemas CSCL.

2.2. Sistemas CSCL

Las siglas CSCL corresponden a "*Computer Supported Collaborative Learning*" y se refieren al aprendizaje a través de la realización de tareas grupales. Es importante diferenciar el aprendizaje individual del aprendizaje realizado en grupo. En el individual uno depende de sus propios medios solamente. Sin embargo, en el aprendizaje grupal se puede recibir ayuda o ayudar a otros a entender mejor un concepto creando enlaces y habilidades sociales además de conocimiento.

Jugar es una muy buena forma de aprender en grupo y en este aspecto existen juegos multiusuario que apoyan el aprendizaje de cietrtas habilidades (memorización, coordinación psicomotora, etc.) o conocimiento.

El aprendizaje apoyado por ordenador puede llevarse en el aula pero también en casa o a distancia, lo cual se conoce como "*e-learning*" o *aprendizaje en línea* [Rosenberg, 2001]. Las plataformas para soportar el *e-learning* pueden apoyar el trabajo en grupo o simplemente no hacerlo. Si lo hace, deberá soportar la comunicación entre usuarios y la realización de actividades colaborativas y/o cooperativas, siendo como principal objetivo la generación de conocimiento y la compartición del mismo.

Una plataforma para el aprendizaje colaborativo es Elluminate[7], la cual proporciona un espacio compartido en el que se van mostrando texto, diagramas o dibujos, y en el que pueden comunicarse los usuarios entre sí y pueden recibir ayuda del profesor o profesores (Figura 3[8]).

En general, los sistemas CSCL deben soportar los aspectos generales del Groupware y los mecanismos necesarios para generar, adquirir y compartir conocimiento.

[7] `www.elluminate.com/`

[8] `http://sheppartoncrtsupportnetwork.blogspot.com.es/2010/06/virtual-conferencing-with-elluminate.html`

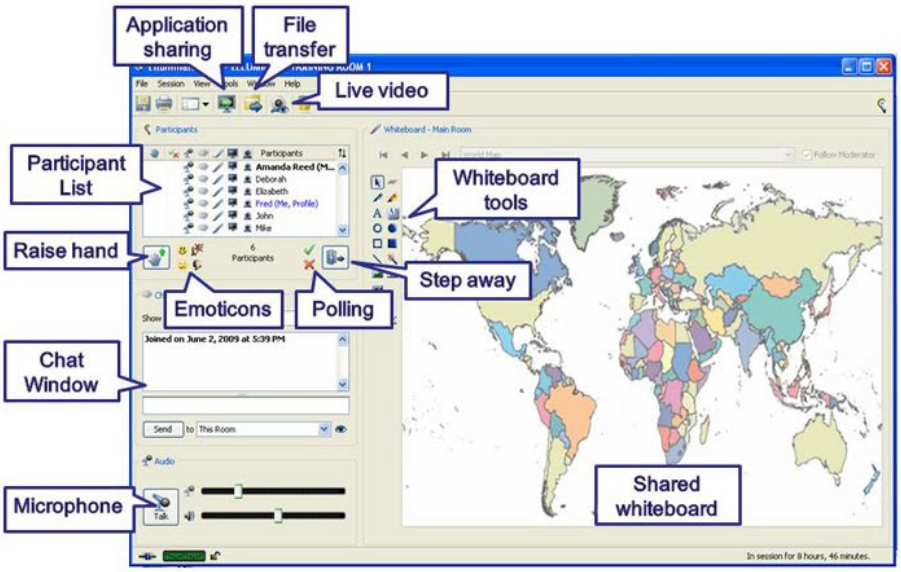


Figura 3. Componentes de la interfaz gráfica del software Elluminate, el cual soporta el aprendizaje colaborativo en línea.

2.3. Juegos y redes sociales

En el área de los juegos, algunos pueden llegar a tener miles de usuarios conectados a la vez, como es el caso de "*The World of Warcraft*"[9] (WoW), el cual es un juego de rol en línea multijugador masivo[10].

Un juego multijugador implica varios jugadores a la vez jugando normalmente en un espacio que puede ser compartido, como una arena o un escenario. Los roles implican diferentes características, habilidades y objetivos en el juego. La parte *en-línea* significa que los juegos requieren acceso a Internet. Son masivos por soportar miles de usuarios a la vez y ciertamente no es lo mismo jugar con una o dos personas a jugar con unas cuantas miles por ahí.

Otros juegos de estrategia como el "*Age of Empires*" o el ya adulto "*StarCraft*" también utilizan una arena compartida y distintos tipos de jugadores, y aunque la cantidad de jugadores en un escenario no es tan grande, cada jugador controla varios tipos de entidades del juego, como tanques, arqueros, constructores,

[9] `http://en.wikipedia.org/wiki/World_of_warcraft`
[10] `http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game`

carpinteros, etc.

En estos juegos hay mucha información transmitiéndose constantemente a través de la interfaz visual. El escenario, los jugadores, las entidades del juego y los datos en tiempo real del estado de cada una de las entidades del juego. Por ejemplo, puedo tener 20 tanques pero también puedo querer saber cual es el estado de un solo tanque.

Sobra decir que el propósito de estos juegos es simplemente divertirse con otras personas. Sin embargo, sus características los hacen ser sistemas colaborativos/cooperativos muy complejos.

Las redes sociales por otra parte permiten que los usuarios formen grupos y que en esos grupos compartan mensajes, imágenes, videos, documentos, enlaces a sitios de interés, etc. Son plataformas para compartir lo que a uno se le ocurra. ¿Por qué son tan populares las redes sociales? Porque cualquier persona con acceso a Internet puede reencontrarse con amigos y familiares, a la vez que conoce *virtualmente* a otras personas, ya sea con amigos o familiares en común o con ideas y gustos en común. Existen redes sociales especializadas como Last.fm[11], la cual permite compartir gustos musicales. Por ejemplo, si una persona escucha música similar a la que tu escuchas, esta red puede recomendarte la música que escucha esa persona con gustos similares.

En cierta manera, una red social es un sistema Groupware en la que los usuarios crean recursos compartidos y comunicaciones referentes a los mismos, dependiendo si la red social es de propósito general o especializada para compartir conocimiento científico, contactos empresariales, o música.

Otros sistemas con usos muy diversos son los sistemas de mensajería instánea, los cuales son por sí mismos una categoría propia de sistemas Groupware, ya que su único propósito es permitir la comunicación escrita y tal vez oral entre uno o varios usuarios. De nuevo, un usuario contacta con otro usuario a través de mensajes o voz. El sistema de correos permite esa comunicación pero de

[11] `www.last.fm`

manera asíncrona, mientras que los sistemas de mensajería instantánea lo hacen en tiempo real y se utilizan para el mismo propśito: la comunicación en tiempo real.

3. Metodologías

La complejidad de los sistemas Groupware ha hecho que se creen metodologías especializadas que toman en cuenta las dificiultades específicas de este tipo de sistemas, los cuales se identifican por soportar tareas colaborativas y/o cooperativas (Ver [Dillenbourg et al., 1996] para las diferencias entre colaboración y cooperación).

Consideramos que el Awareness es parte del proceso de interacción y por lo tanto, estará presente en los procesos interactivos más complejos como la colaboración y la cooperación.

En esta sección describimos de forma general varias metodologías de desarrollo especializadas en los sistemas Groupware, las cuales utilizan diferentes enfoques para su proceso de desarrollo y para la base conceptual que soporta cada una.

El objetivo de esta sección es detectar en cada una de ellas el soporte del Awareness, ya sea implícita o explícitamente, a fin de saber a que nivel está integrado el Awareness a las metodologías de desarrollo revisadas.

Las metodologías revisadas son AMENITIES [Garrido et al., 2002], TOUCHÉ[12] [Penichet et al., 2010] y CIAM [Molina et al., 2009].

3.1. AMENITIES

AMENITIES (acrónimo de **A** **ME**todology for A**N**alysis and Des**I**gn of coopera**TI**ve syst**E**m**S**) [Garrido et al., 2002] es una metodología para el análisis y diseño de sistemas cooperativos. La metodología permite realizar un modelado

[12] `http://www.penichet.net/`

conceptual del sistema cooperativo; centrándose en el grupo y cubriendo aspectos relevantes de su comportamiento (dinámicas, evolución, etc.) y estructura (organización, leyes, etc.).

El objetivo de la metodología es abordar de forma sistemática el análisis y diseño de sistemas cooperativos facilitando el desarrollo posterior del software. AMENITIES proporciona un marco de referencia conceptual y metodológico, pero además propone una metodología concreta.

En particular aporta los siguientes elementos:

1. **Fases generales** para analizar y diseñar sistemas cooperativos.

2. **Fases específicas** para la construcción de los diferentes modelos implicados en la metodología.

3. La definición de un **marco conceptual de trabajo** para dar soporte al modelado de estos sistemas.

4. Una **notación** denominada COMO-UML (acrónimo de COOperative MOdel notation based on UML) que se basa en el lenguaje UML [Group, 2001], el cual se considera el lenguaje estándar de facto para especificar y construir sistemas software. La notación se utiliza para la construcción del modelo (denominado modelo cooperativo) considerado el núcleo central de la metodología.

5. Una semántica precisa de la notación anterior en el dominio del problema (sistemas cooperativos) mediante el formalismo de las Redes de Petri (CPN-Coloured Petri Nets) [Jensen, 1996].

6. Una herramienta denominada COMO-TOOL (acrónimo de COOperative MOdel TOOL) con diferentes funciones de apoyo al análisis y modelado de estos sistemas.

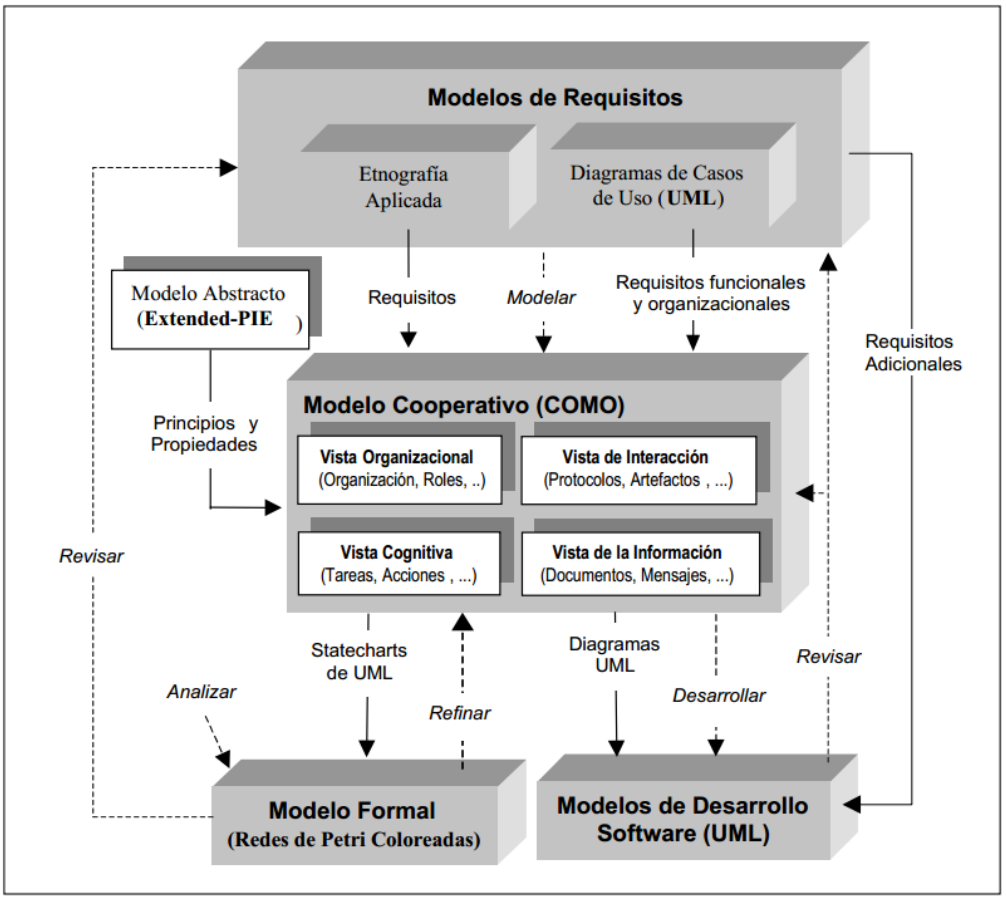


Figura 4. Esquema general de la metodología AMENITIES incluyendo las vistas del modelo cooperativo.

3.1.1. *Esquema general*

La metodología AMENITIES está compuesta por componentes abstractos, descriptivos y formales, según se muestra en la Figura 4.

Las fases generales de la metodología son las siguientes:

1. Análisis del sistema y obtención de requisitos.
2. Modelado del sistema cooperativo.
3. Análisis del modelo cooperativo.
4. Diseño del sistema, introduciendo nuevos elementos o cambios en el modelo

cooperativo, probablemente como resultado del análisis anterior.

5. Desarrollo del sistema software.

Como en la mayoría de las metodologías, la aplicación de AMENITIES a un sistema sigue un proceso iterativo simple, permitiendo llevar a cabo un refinamiento del modelo como consecuencia del análisis de éste, así como una revisión de los requisitos de partida o del modelo cooperativo que podría aportar nueva o diferente información a considerar.

3.1.2. *Modelo de requisitos*

La obtención y especificación de los requisitos se lleva a cabo mediante la técnica de la etnografía aplicada y de los diagrams de casos de uso UML. En primer lugar, la aplicación de la etnografía aplicada y permite reunir informalmente los requisitos del sistema a estudiar y desarrollar. A continuación, a partir de esta información, los diagramas de casos de uso se utilizan para estructurar y especificar los requisitos funcionales, roles y actores del sistema.

La etnografía permite identificar:

- La **estructura de los grupos** en base a ciertos aspectos en muchos casos sutiles pero de vital importancia (comportamiento del participante dentro y fuera del grupo, responsabilidades directas y asumidas, etc.).
- Cuestiones socioculturales que pueden afectar a la **interacción y comunicación** entre participantes para lo cual utilizan artefactos tales como videoconfencia por computadora, editores o espacios de trabajo compartidos, etc.
- **Comportamientos** de los participantes y las posibles influencias de este comportamiento en el resto de participantes que forman parte de un grupo.
- Requisitos funcionales relacionados con las **prácticas de trabajo** y con los **recursos** necesarios para llevar a cabo el trabajo en grupo.

Los casos de uso [Rumbaugh et al., 1999] permiten identificar requisitos funcionales acerca del comportamiento del sistema y por tanto representar de forma abstracta los objetivos de los usuario. Un caso de uso es una unidad coherente de funcionalidad expresada como una transacción entre los usuarios y el sistema.

Los diagramas de casos de uso se utilizan en AMENITIES para representar (mediante una notación semi-formal) parte de la información recabada mediante la etnografía aplicada:

- La **funcionalidad del sistema** según se percibe por los usuarios, incluyendo relaciones básicas (generalización o especialización, extensión e inclusión) entre estas funciones.
- **Identificación y clasificación de los actores** que utilizan y/o realizan casos de uso, también incluyendo relaciones básicas (especialización o generalización) entre ellos si las hay.

3.1.3. *Modelo cooperativo*

El modelo cooperativo es el núcleo central de AMENITIES. Es un modelo conceptual formado por un conjunto de modelos de comportamiento y tareas, los cuales son de utilidad para describir y representar cualquier sistema cooperativo desde el punto de vista de su estructura y funcionamiento. Su propósito es dar una descripción del sistema independientemente de su implementación, proporcionando así una mejor comprensión del dominio del problema, dada la complejidad que caracteriza a este tipo de sistemas.

El modelo cooperativo se compone de cuatro vistas conceptuales (Figura 4): La vista organizacional, la vista cognitiva, la vista de interacción y la vista de información.

La vista organizacional permite describir cómo se estructuran los grupos que forman parte de las organizaciones, en concreto, aspectos tales como: grado de acoplamiento de los participantes, sus cargas de trabajo, la evolución del grupo,

etc. AMENITIES permite especificar grupos de la organización y equipos de trabajo.

La vista cognitiva fundamentalmente representa el conocimiento de cada miembro del grupo en el contexto organizacional, tanto el conocimiento adquirido como el que se requiere. Tradicionalmente, el conocimiento que las personas tienen acerca de las tareas que pueden realizar, es un subconjunto importante del conocimiento total que ellos poseen. Los usuarios recurrirán a ese conocimiento cuando usen el software para realizar las tareas [Johnson, 1989].

La vista cognitiva aborda dos temas:

- Establece enlaces entre el modelado de grupos de usuarios y el modelado de tareas estructuradas.
- Sienta las bases para estudiar y analizar cómo el diseño del sistema Groupware influye en la usabilidad del sistema, por supuesto, en términos de comportamiento de los usuarios y tareas que realizan en grupo.
- Representación de consideraciones sociales y personales en relación con otros conceptos que forman parte del escenario compartido: circunstancias en las cuales los actores pueden desempeñar roles, posibles restricciones, etc.
- Especificación de estrategias a la hora de llevar a cabo tareas, así como especificar excepciones para circunstancias que las puedan requerir.

La interacción es un proceso natural en el grupo, ya sea mediante la participación en tareas comunes, o cuando las acciones de un miembro en el sistema afectan directa o indirectamente al comportamiento de otros miembros o grupos (restricciones en cambios de rol social, etc.). La vista de interacción hace referencia a la interacción explícita entre participantes cuando no es posible garantizar que se cumplan las reglas o pasos que gobiernan el propio proceso de

interacción, es decir, casos en los cuales las tareas son débilmente estructuradas o sin estructurar.

El objetivo de la vista de interacción en el modelo cooperativo consiste en representar algunos o todos los siguientes aspectos que se hayan podido identificar para una actividad que requiere interacción directa entre participantes:

- La forma de llevarla a cabo, sus restricciones temporales (síncrona o asíncrona) y el marco de tiempo (largo, medio y corto).
- Los participantes (humanos, agentes, etc.) y sus grados de cooperación.
- Uso de protocolos sociales para alcanzar el objetivo.
- Los requisitos de soporte a la comunicación entre participantes, y a la coordinación (controlada o guiada) de las acciones que deben llevar a cabo. Se tienen en cuenta factores relevantes relacionados con los artefactos utilizados (si los hay) en el proceso de interacción ya que la interacción ocurre a través de algún medio (cara a cara, redes de computadoras, etc.).

La vista de la información es la encargada de representar datos tales como documentos y mensajes, cuya principal finalidad es comunicar información de una manera más o menos estructurada. Esto permite representar dentro del modelo:

- Información abstracta(a veces también denominada sin designación) como pueden ser los mensajes.
- Información concreta (o con designación) como pueden ser documentos o datos multimedia.

Existen modelos que consideran los datos como el centro de toda colaboración/interacción que se produce en el sistema. Desde este punto de vista, todos los datos necesitan ser entidades tangibles. Por el contrario, según el nivel de

abstracción que proporciona el modelo cooperativo de AMENITIES, la información es un elemento más del sistema que puede estar implícita (por ejemplo los mensajes) en actividades o acciones, y si se cree conveniente también se puede hacer explícita como flujo de información entre éstas. De esta manera, los datos y objetos se podrán identificar y manejar con miras al proceso global.

Hay información implícita (que normalmente forma parte de actos de interacción entre humanos) difícil de representar en un modelo. Es el caso por ejemplo de los gestos. A pesar de no ser posible su representación, el modelo cooperativo sí permite representar, mediante la vista de la interacción descrita anteriormente el soporte necesario para que los participantes puedan proporcionar y utilizar esta información. Así es posible prever, en la implementación o durante el funcionamiento del sistema, el conjunto de medios (videoconferencia, etc.) que cumplen los requisitos de soporte para las características especificadas. En este sentido, la vista de la información permite hacer explícita parte del contexto de sistema según se requiera.

La vista de la informacón necesita tanto de la vista cognitiva como de la vista de interacción puesto que las entidades que representa (documentos, mensajes, etc.) están ligadas directamente a las subactividades/acciones de cualquier tipo que se realizan en el sistema cooperativo. Por ejemplo, un mensaje entre dos usuarios o entre un usuario y la aplicación tiene sentido a no ser que esté ligado a algún proceso de colaboración concreta.

3.1.4. *Marco conceptual de trabajo*

La base sobre la cual se realiza el proceso de modelado es el marco conceptual de AMENITIES.

Ciertos conceptos presentes en modelos sociales no son fáciles de encajar en modelos computacionales debido a su nivel de abstracción [Wood et al., 1998]. El modelo del sistema cooperativo de AMENITIES intenta solucionar este inconveniente sirviendo como paso previo a los modelos utilizados en el desarrollo del

sistema. Para lo cual definimos los conceptos que consideramos más relevantes en un sistema cooperativo tal como se muestra en la Tabla 1:

Las relaciones entre los conceptos del modelo conceptual de AMENITIES se muestran en la Figura 5.

Como puede observarse en dicha figura, el sistema cooperativo está formado a un primer nivel por:

- Una o más organizaciones estructuradas cada una de ellas en base a un conjunto de suborganizaciones o roles relacionados.
- Uno o más grupos de actores.
- Leyes que establecen las reglas básicas de comportamiento del grupo dentro de las organizaciones.
- Eventos en el sistema.
- Artefactos disponibles.
- Información.

A un segundo nivel:

- Los roles incluyen tareas, y éstas están formadas por unidades de trabajo, es decir, por subactividades, y las subactividades por otras subactividades y/o acciones.
- Los grupos están formados por actores los cuales tienen adquiridas una serie de capacidades.

Las cuatro etapas para describir el modelo cooperativo son:

1. Se especifica la **organización** mediante un conjunto de roles conectados por transiciones.

Concepto	Definición
Evento	Ocurrencia de algún hecho que tiene localización tanto en el espacio como en el tiempo.
Acción	Unidad básica de trabajo ejecutable atómicamente.
Artefacto	Dispositivo (hardware y/o aplicación software) utilizado para llevar a cabo ciertas acciones.
Objeto de información	Entidad que contiene la información requerida para llevar a cabo acciones, o que se genera como resultado de éstas.
Subactividad	Unidad de trabajo formada por un conjunto de acciones y otras subactividades que permite estructurar tareas.
Tarea	Conjunto de subacciones/acciones cuya realización permite alcanzar objetivos.
Actor	Usuario, programa o entidad que puede desempeñar roles.
Rol	Comportamiento esperado de un actor en base a un conjunto identificable de tareas a realizar.
Capacidad	Habilidad o responsabilidad asociada a un actor que le permite desempeñar roles y llevar a cabo tareas, subactividades o acciones.
Tarea cooperativa	Tarea en la cual para su realización interviene más de un actor, bien desempeñando el mismo rol o distintos.
Organización	Conjunto de roles, y relaciones entre ellos, que se dan en un lugar.
Ley	Norma impuesta por una organización que restringe su funcionamiento en base a reglas sociales, culturales, capacidades de los actores, etc.
Grupo	Conjunto de actores desempeñando roles que pertenecen a una misma organización o que participan en la realización de tareas cooperativas.
Protocolo de interacción	Conjunto de reglas de comportamiento que utilizan los miembros de un grupo para llevar a cabo subactividades.

Tabla 1. Definición de conceptos para el marco de trabajo del modelo cooperativo de AMENITIES [Garrido Bullejos, 2003]

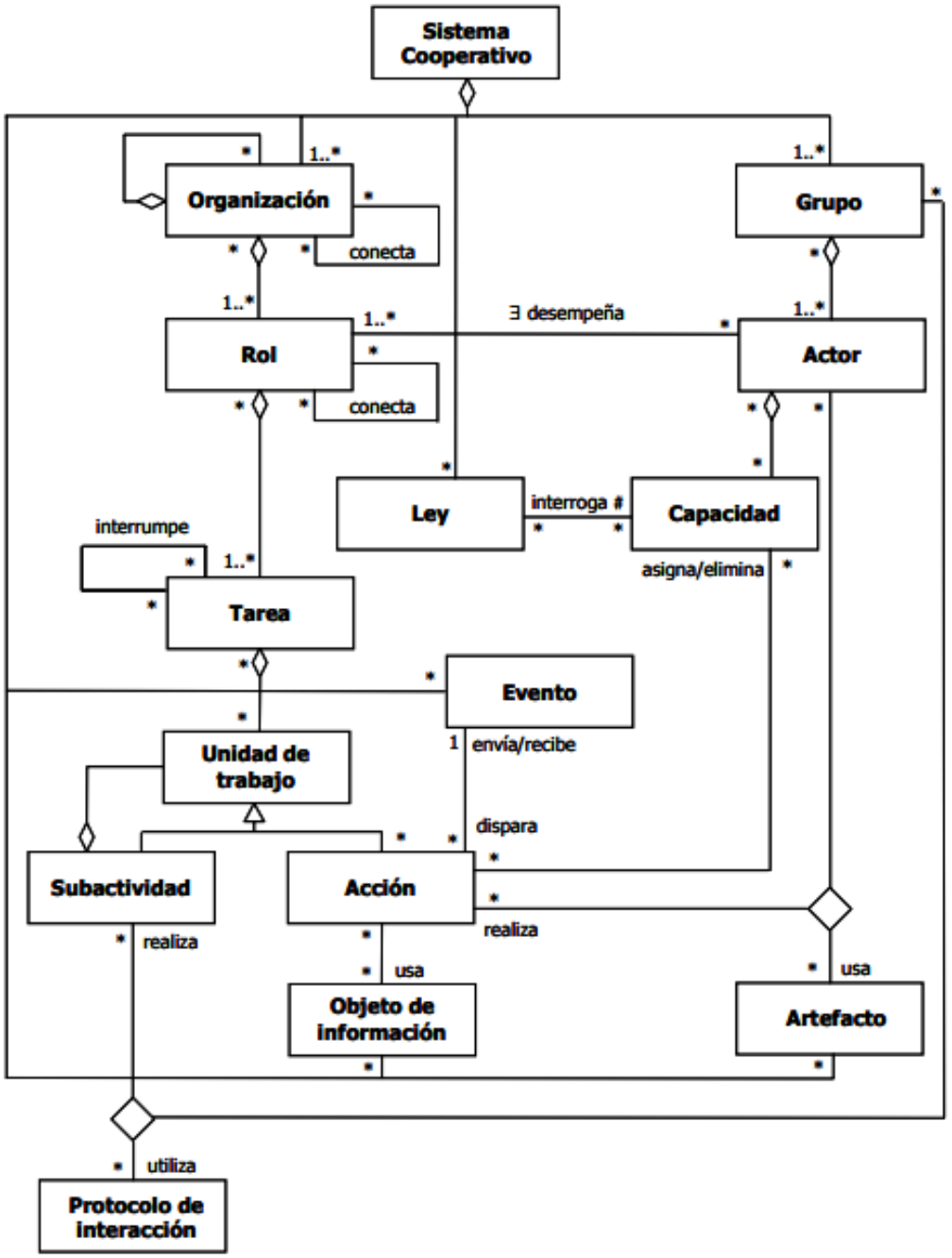


Figura 5. Relaciones del marco conceptual de trabajo de AMENITIES [Garrido Bullejos, 2003]

.

2. Se definen cada uno de los **roles** en base a conjuntos de tareas.
3. Se definen las **tareas** mediante conjuntos de subactividades y acciones.
4. Se especifican protocolos de interacción para aquellas subactividades que lo requieran.

Para especificar la organización se utilizan los llamados **diagramas de organización** basados en los diagramas de estado UML. Se pueden especificar los grupos de la organización y los equipos de trabajo.

La definición de los roles (como parte de la vista cognitiva) sirve de enlace entre la especificación del comportamiento de los grupos de la organización y las actividades individuales y/o colaborativas a realizar por los miembros del grupo.

El concepto clave sobre el que se articula la conexión entre un grupo de la organización y el trabajo a realizar por éste, es el de **tarea**. La estrategia a seguir para definir los roles consiste en identificar y asociar las tareas a roles partiendo de la información, referente al comportamiento social, contenida en los modelos de requisitos de AMENITIES (etnografía aplicada y diagramas de casos de uso UML).

Para cada tarea en cada rol podemos espeficiar lo siguiente:

- Un conjunto de **tareas** que pueden interrumpir su realización. Estas tareas pueden pertenecer al mismo rol o a otros. Por defecto, la realización de una tarea no puede ser interrumpida.
- **Eventos** que disparan la realización de la tarea.
- **Leyes** que habilitan la realización de la tarea por miembros del grupo.
- **Acciones** en las transiciones de entrada y salida a la tarea, las cuales pueden llevar a cabo cambios controlados en el estado global del sistema. Por ejemplo, contabilizando el número de participantes que están colaborando actualmente en la realización de una tarea. Estos cambios pueden

repercutir directamente en que se satisfagan leyes según los términos en los cuales éstas están expresadas. Por ejemplo, podemos limitar el número de participantes en una tarea mediante una ley que tenga en cuenta el estado del sistema en relación con el número de participantes que colaboran actualmente en dicha tarea.

- **Objetos de información** requeridos para llevar a cabo la tarea. Estos objetos proporcionan parte de la vista de la información ya que también se pueden incluir objetos de información las etapas siguientes.

- **Tipo de tarea** en base a si son necesarios varios actores para llevarla a cabo (individual o cooperativa).

- En caso de ser una tarea cooperativa, hay que especificar la multiplicidad, es decir, el número de actores que han de intervenir en la realización de dicha tarea bajo cada uno de los roles implicados.

La definición de las tareas se realiza mediante los diagramas de tareas y subactividades en notación COMO-UML, los cuales están basados en los diagramas de tareas de UML.

Los protocolos de interacción pueden definirse usando diagramas de subactividades en COMO-UML. La idea detrás de los protocolos de interacción es poder especificar propiedades que caracterizan las interacciones entre participantes en base a aspectos tan diversos como protocolos sociales, patrones de comportamiento, tecnologías, etc. Los protocolos de interacción que aquí se especifican y definen se asocian a subactividades que se llevan a cabo mediante la interacción directa entre participantes. Así se puede proporcionar el soporte adecuado y necesario mediante protocolos tecnológicos implícitos en artefactos (por ejemplo, el correo electrónico) que garanticen la realización de la subactividad conforme a los protocolos sociales que rigen la coordinación de las acciones que forman parte de dicha subactividad.

La especificación de un protocolo de interacción para llevar a cabo una subactividad incluye la siguiente información:

1. Los roles participantes que tienen responsabilidades en la subactividad.
2. El tipo de reglas que guían la interacción.
3. Cualquier otro elemento necesario tal como: requisitos de comunicación específicos que sustituyan a los heredados, restricción o ley.

3.1.5. *Modelos de desarrollo del software*

Para conectar con el desarrollo del software, AMENITIES propone hacer uso de las notaciones disponibles en UML. El uso de la notación COMO-UML para la construcción del modelo cooperativo, facilitarán en gran medida esta fase de desarrollo.

Los modelos dinámicos que forman parte del modelo cooperativo están directamente relacionados con el desarrollo de las aplicaciones, en concreto, con modelos arquitectónicos, funcionales y de comportamiento de los distintos componentes que van a formar parte de la aplicación.

El análisis y modelado de tareas se puede utilizar para dirigir el desarrollo del software de varias formas (análisis de requisitos, derivar interfaces de usuario, evaluaciónes, etc.). En AMENITIES, la principal diferencia radica en que el modelo cooperativo que sirve como punto de partida para el desarrollo del software, no sólo describe tareas, sino que además abarca e integra junto con éstas otros conceptos tales como: organización, roles, etc. De forma análoga a la utilización tradicional de modelos de tareas, la idea es poder derivar por ejemplo, de forma al menos semiautomática, la interfaz de usuario de las aplicaciones groupware [Paterno, 2000].

El punto de conexión de AMENITIES con el desarrollo del software es el diseño arquitectónico, el cual va a proporcionar una visión global común a cada

miembro del equipo de desarrollo. Básicamente, un diseño arquitectónico del software proporciona un conjunto de componentes y de relaciones entre ellos.

Es de vital importancia la implementación del sistema a partir de un conjunto de subsistemas que se comunican a través de interfaces bien definidas, ya que un sistema cooperativo inherentemente es un sistema distribuido.

Para llevar a cabo el diseño arquitectónico de los sistemas groupware, la metodología AMENITIES pretende hacer uso del lenguaje UML de la siguiente manera:

1. Para representar la descomposición de un sistema en subsistemas más pequeños se van a utilizar los diagramas de componentes, en concreto los **diagramas de paquetes** con los estereotipos "*sistema*" y "*subsistemas*" junto con la relación de **composición** (una forma de la asociación de agregación que considera partes de un todo con tiempos de existencia ligados).

2. Representar la estructura funcional (vista estática) de los subsistemas dentro del diseño arquitectónico mediante **diagramas de clases e interfaces**.

3. Describir el comportamiento (vista dinámica) de los subsistemas que colaboran en base a la estructura funcional descrita en el punto anterior utilizando **diagramas de colaboración**.

4. Finalmente, describir la distribución de los subsistemas a desarrollar en los nodos del sistema distribuido mediante **diagramas de componentes y despliegue**.

3.2. TOUCHÉ

TOUCHÉ [Penichet et al., 2009b], es un modelo de proceso y una metodología para el desarrollo de interfaces de usuario para aplicaciones groupware desde la elicitación de requisitos hasta su implementación considerando las características y particularidades de estos sistemas desde el inicio.

El nombre "TOUCHÉ" viene de Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments o lo que es lo mismo Modelo de Proceso para el Desarrollo de Interfaces en Entornos CSCW Centrado en los Usuarios y Dirigido por Tareas.

TOUCHÉ es un modelo de proceso y metodología de desarrollo a la vez. Se trata de un modelo de proceso puesto que indica las etapas por las que se ha de seguir desde el inicio, desde el estudio de los requisitos del sistema, hasta la implementación de la interfaz de usuario.

Por otro lado, se trata también de una metodología puesto que indica, en cada una de esas etapas, lo que se ha de hacer para modelar el sistema a desarrollar: se proponen una serie de artefactos (diagramas, plantillas, etc.) para modelar la realidad de un modo formal o semi-formal de manera que los artefactos guarden una relación entre sí y se mantenga, además, una coherencia entre ellos (trazabilidad).

El número de técnicas a emplear en las etapas del modelo de proceso puede ser muy numeroso. No se descarta la posibilidad de emplear cualquier otra técnica, no contemplada en esta metodología, que pudiera complementar la información de la especificación, pero esa posibilidad, ese gran número de técnicas, podría llegar a complicar en exceso la descripción del sistema.

Si un analista se encuentra con una serie de pasos y técnicas a seguir, que le resulten más o menos viables por sencillez y tiempo, es posible que la especificación que haga sea más exitosa que si encuentra un número excesivo de posibilidades que le hagan "perderse en la abundancia".

Por este motivo se concretan las técnicas a emplear en la metodología, sugiriendo aquellas que se consideran oportunas de entre las que son ampliamente utilizadas y aceptadas. En otras ocasiones, se extienden, se modifican o se crean nuevas técnicas que permitan mejorar y completar la especificación, normalmente aportando una visión colaborativa.

El modelo de proceso desarrollado para el estudio, anásis y diseño entornos colaborativos consta de cuatro pasos interrelacionados y dependientes. Por un lado son independientes porque en cada etapa se abordan asuntos concretos y específicos correspondientes a dicha etapa y no a otra.

Sin embargo, el análisis y diseño de los sistemas informáticos en general, no sólo de entornos colaborativos, hace imposible una independencia absoluta entre etapas. Existe pues una relación entre ellas de manera que se pueda pasar de una etapa a otra o de manera que un cambio en una etapa puede tener una repercusión en otras.

Por lo tanto la trazabilidad entre las etapas identificadas es tan importante como cada una de las etapas para mantener la coherencia del modelo y de la especificación del sistema.

El modelo de procesos desarrollado en TOUCHÉ es una combinación entre un proceso de desarrollo tradicional de Ingeniería del Software y uno de Interacción Persona-Ordenador. En el primero se suele tener en cuenta la etapa de requisitos, mientras que en el segundo se suele comenzar desde el anásis de las tareas. El punto clave en esta "unión" ha sido la relación "Requisito - Tarea". Las etapas del modelo de proceso son las siguientes (ver Figura 6):

- Elicitación de Requisitos : La primera etapa del modelo reúne los requerimientos del sistema a ser desarrollado para cumplir los objetivos de los usuarios. Esta parte se basa en el trabajo de Amador Durán, donde es importante remarcar, además de otras cosas, las plantillas para la obtención de requerimientos y el Documento de Requerimiento de Sistema, el cual reúne toda la información.

- Análisis: La etapa del Análisis se trata del estudio del dominio. Describe los requerimientos del sistema sin describir las cuestiones de la implementación. Los roles y las tareas son identificados y descritos. El análisis del sistema se hace desde una perspectiva estructural a través de Diagramas

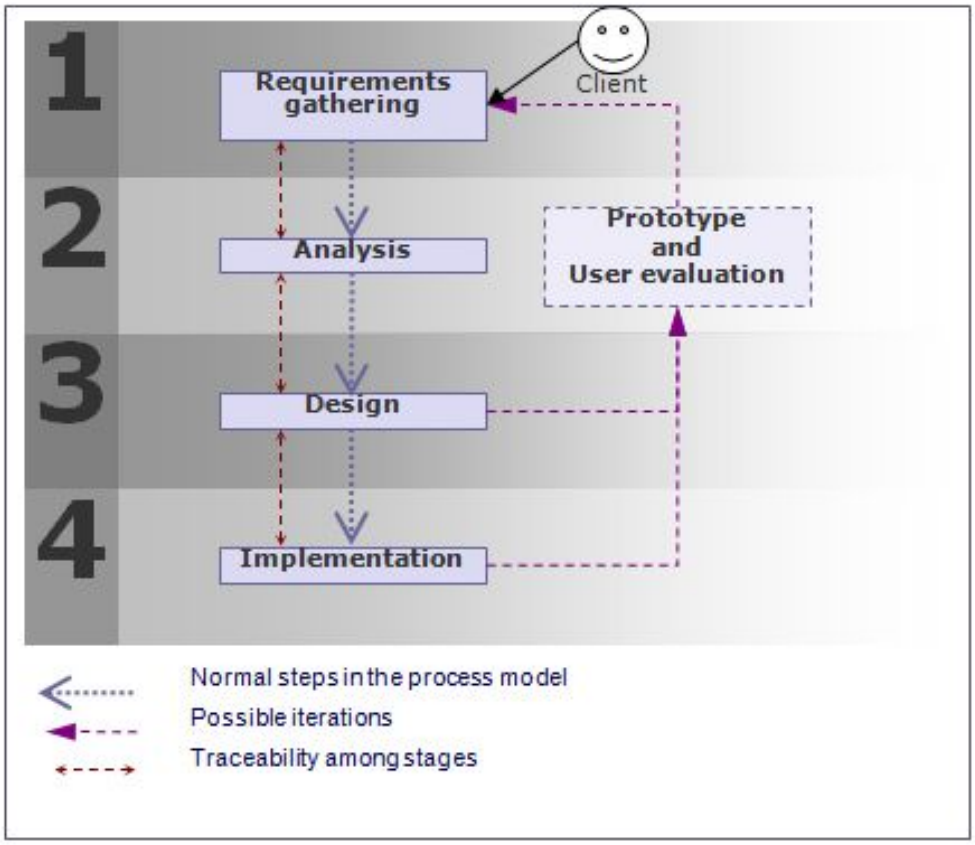


Figura 6. El flujo de trabajo de la metodología TOUCHÉ.

de Clases y del Diagrama de la Estructura Organizacional; y desde la perspectiva del comportamiento a través del Diagrama de Tareas (La notación CTT se ha sido adoptada en este caso) y del Diagrama de Co-Interacción (desarrollado para identificar relaciones entre los actores del sistema).

- Diseño: Esta etapa trata sobre cómo representar la información (visualización, entradas, controles, etc.) a los usuarios finales. Toda la información reunida hasta ahora en las etapas previas es procesada y transformada a una representación manejable y ejecutable por software. El "*Awareness del Usuario*", el cual se refiere al conocimiento acerca de los otros usuarios del sistema, debería ser considerado a fin de obtener un buen Groupware. Los Objetos Abstractos de Interacción (AIO) son usados para diseñar Interfaces de Usuario Abstractas. Utiliza un esquema conceptual de UsiXML [Limbourg et al., 2005] para la especificación de las interfaces de usuario. El modelo es enriquecido con un nuevo AIO y nuevas facetas las cuales proporcionan mayor expresividad para representar este tipo de sistemas.
- Implementación: La etapa de Implementación del sistema trata sobre la generación de las interfaces de usuario de los AIOs obtenidos en la etapa previa. Es un proceso de reificación de cada componente a elementos más concretos (Objetos Concretos de Interacción o CIO) de acuerdo con la implementación y los detalles de la plataforma. Uno de los últimos estudios sobre interfaces de usuario para entornos HCI es Cameleon [Calvary et al., 2003]; este trabajo sigue dicho marco de referencia. Varios CIOs específicos para el Groupware también son agregados.

Este modelo de proceso incluye las prácticas y estrategias que se consideran necesarias para mejorar el proceso de desarrollo. Como prácticamente todos los modelos empleados en Ingeniería del Software también es iterativo, lo que permite refinar cada uno de los pasos hasta obtener el producto final deseado.

Tras el diseño se considera la realización de un prototipo que permita la evaluación de los usuarios. Esto permite que sean ellos los que validen como va

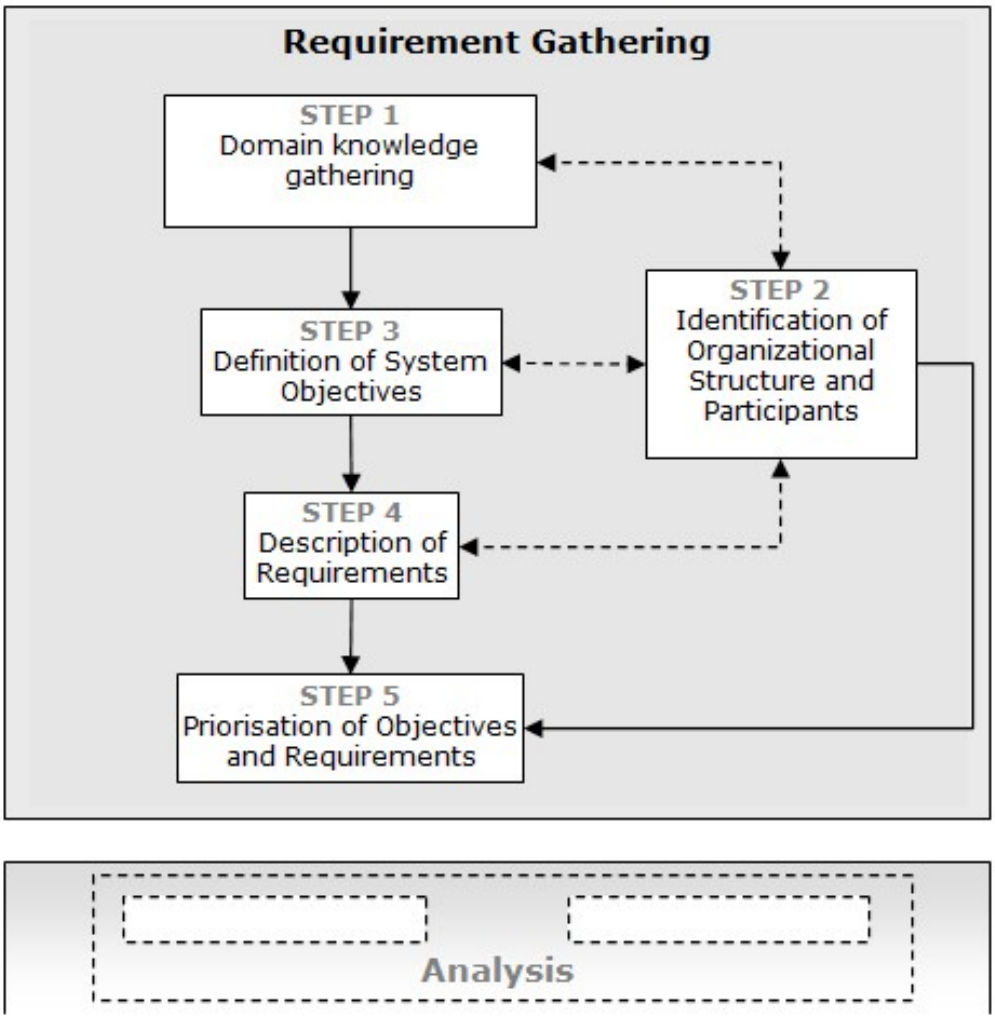


Figura 7. Proceso de elicitación de requisitos en TOUCHÉ

la aplicación, dándoles mayor participación y mayor valor a su opinión.

A continuación se describen cada una de las etapas de la metodología TOUCHÉ.

3.2.1. *Elicitación de requisitos*

La elicitación de requisitos en TOUCHÉ es la primera fase de la metodología, la cual consta de los siguientes pasos, también mostrados en la Figura 7:

1. **Adquisición del conocimiento del dominio del problema**. Este paso comprende la recopilación de toda la documentación posible sobre el dominio del problema y su entorno. Esto incluye documentación diversa, entrevistas, cuestionarios, inmersión en el negocio del cliente, modelado estructural del sistema, etc.

2. **Identificar la estructura organizativa y los actores del sistema**. Este paso comprende la identifcación de cada rol en la organización, así como cada tipo de usuario posible, ya sea de forma individual o como un grupo determinado. Las relaciones entre cada uno de los actores en el sistema también son importantes, ya que de ahí se obtiene la estructura jerárquica, de mando o de responsabilidades. Este paso es paralelo al 1, 3 y 4, ya que se irá enriqueciendo conforme se avance en la elicitación de requisitos. Para esta etapa se utilizan las plantillas para la Estructura Organizativa y las extensiones específicias para grupos, individuos, usuarios y agentes [Ruiz Penichet, 2007].

3. **Elicitar los objetivos del sistema**. En este paso se busca identificar de forma explícita cada uno de los objetivos del sistema. Se utilizan plantillas como la mostrada en la Tabla 2, la cual está enfocada a la definición de objetivos o requerimientos del sistema.

4. **Elicitar los requisitos**. La elicitación de requisitos de información funcionales y no funcionales también se realiza a partir del estudio y análisis de la información obtenida en las etapas anteriores.

   Se identifican los requisitos de información relativos al almacenamiento de información que deberá cumplir el sistema software a desarrollar; los requisitos funcionales, (casos de uso) que deberá cumplir el sistema software a desarrollar; y los requisitos no funcionales, (normalmente de carácter teórico o legal) que deberá cumplir el sistema software a desarrollar. Todo ello teniendo en cuenta los metadatos incluidos en las plantillas y la extensión CSCW en caso de que sea necesaria.

5. **Priorizar los objetivos y requisitos**. Se trata únicamente de ordenar por prioridad los objetivos del sistema y los requisitos.

3.2.2. *Análisis*

La etapa de análisis (Figura 8) en las metodologías se corresponde con el estudio del dominio del problema. Esta etapa trata de descubrir el *qué* describiendo los requisitos del sistema sin describir asuntos de implementación.

El análisis de un sistema colaborativo comienza con la identificación previa de roles y tareas a partir de la información recabada en el Documento de Requisitos del Sistema de la etapa anterior. Este comienzo es fruto de la trazabilidad existente entre las etapas de elicitación de requisitos y análisis, que permite identificar roles a partir de actores y tareas a partir de requisitos.

Por un lado, la identificación de los primeros roles permite al analista realizar un primer esbozo de la estructura organizativa de los actores del sistema. Este diagrama, junto con el diagrama de clases de UML modelan la estructura del sistema. Así mismo, la identificación de las primeras tareas permite ordenarlas, creando así un primer esbozo de los diagramas de tareas y los diagramas de co-interacciones, que modelan el comportamiento de los actores del sistema.

Por otro lado, del análisis de las primeras versiones de los diagramas surgen nuevos roles y tareas, es decir, se identifican y describen nuevos roles y tareas como resultado de su estudio.

Tras esa primera identificación de roles y tareas, un proceso iterativo de refinamiento permitirá obtener un modelo de mayor calidad, que permita obtener una especificación completa del sistema.

Del mismo modo que la trazabilidad entre las etapas de elicitación de requisitos y análisis permiten obtener especificaciones coherentes y completas, es necesario tener en cuenta una trazabilidad intra-etapa entre los pasos de la misma.

Plantilla de objetivos o requerimientos del sistema	
{OBJ-, RI-<id>, RF-<id>, RNF-<id>}	<Nombre descriptivo del objetivo, requerimiento de información, requerimiento funcional o requerimiento no funcional>
Versión	*<Número de la versión actual>(<fecha de la versión actual>)*
Autores	() ...
Fuentes	() ...
Objetivos asociados	#OBJ-() ...
Requisitos asociados	#{RI-<id>, RF-<id>, RNF- <id>} (<nombre del requisito>) ...
Importancia	<importancia del objetivo o requisito>
Urgencia	<urgencia del objetivo o requisito>
Estado	<estado del objetivo o requisito>
Estabilidad	<estabilidad del objetivo o requisito>
Necesidad de percepción	De este {objetivo, requisito} deberán estar informados los siguientes actores: #{A-<id>, G- <id>, I-<id>, U-<id>, S-<id>} (<nombre del actor>): - Qué - Cómo - Cuándo - Dónde - Por qué - ...
Participantes	Los actores que participan en la consecución del {objetivo, requisito} son: #{A-, G- , I-, U-, S-} (): ...
Comentarios	<comentarios adicionales sobre la Estructura Organizativa>

Tabla 2. Plantilla general para objetivos y requisitos utilizada en la etapa de Elicitación de Requisitos de TOUCHÉ.

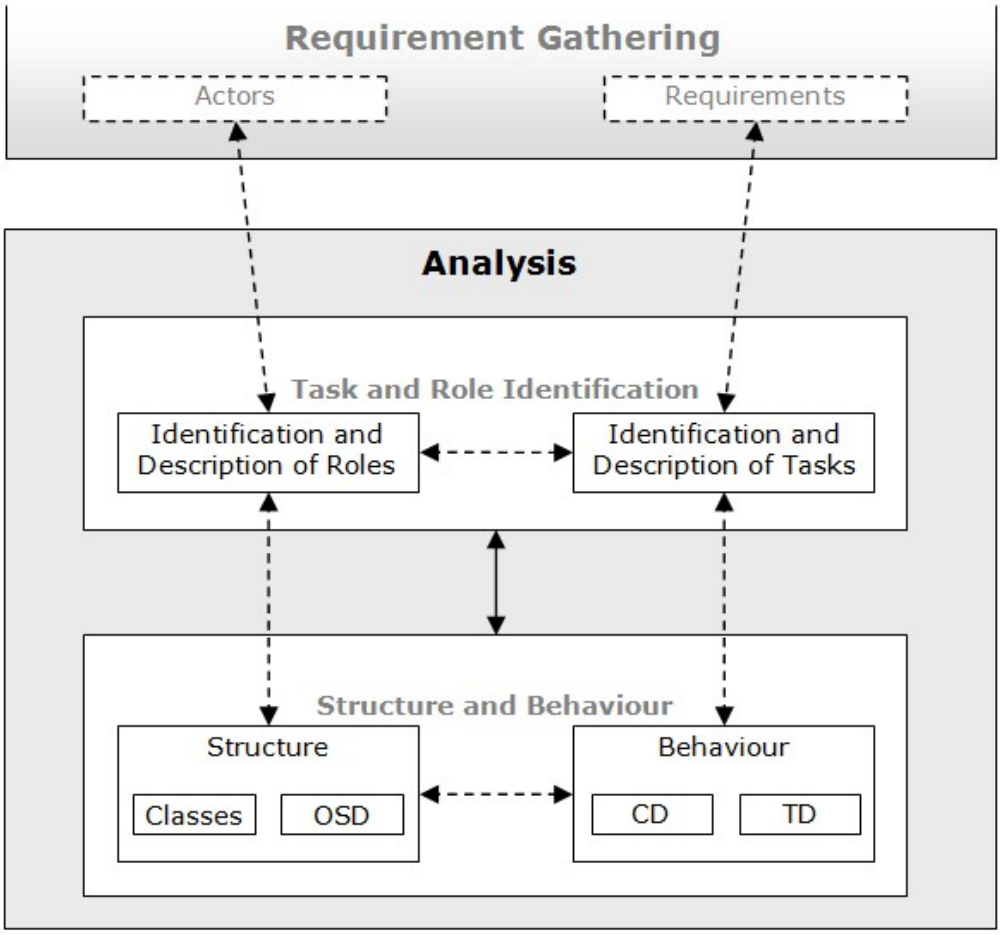


Figura 8. Proceso de elicitación de requisitos en TOUCHÉ

El concepto de rol se ha definido como conjunto de tareas que puede desempeñar un Actor. Los pasos para la identificación y definición de roles son: (1) Seleccionar los actores obtenidos en la fase de la Elicitación de Requerimientos y listar los candidatos, (2) Refinar información e identificar los roles, (3) Describir los roles como parte del modelo organizativo usando usando la plantilla para la Descripción de Roles ([Ruiz Penichet, 2007], pag. 188) y (4) diagramar utilizando el Diagrama de Estructura Organizacional, compuesto por elementos organizativos ([Ruiz Penichet, 2007], pag. 193) y las relaciones entre ellos ([Ruiz Penichet, 2007], pag. 194).

Un diagrama especial se utiliza para representar las colaboraciones entre los actores del sistema. El llamado Diagrama de Co-Interacciones representa las colaboraciones establecidas entre los diferentes elementos organizativos de una estructura organizada, es decir, las interacciones que existen entre los actores (usuarios, agentes, grupos) de un sistema a través del mismo: tareas de grupo.

El Diagrama de Co-Interacciones utiliza los elementos organizativos del sistema y la relación llamada Co-Interacción, la cual es una relación organizativa de grupo entre dos actores que expresa una interacción entre ellos, encaminada al logro de un objetivo común que no podría alcanzarse sin dicha interacción.

La estructura de tareas se representa mediante CTT ([Paternò, 1999] descrita en la Sección 4.3.1), aunque pueden utilizarse otras notaciones.

Para mantener la trazabilidad y la coherencia entre las etapas de elicitación de requisitos y análisis, se establecen dos matrices de trazabilidad ([Ruiz Penichet, 2007], pag. 196) que ponen en relación explícita los actores y los requisitos identificados y descritos en la etapa de elicitación de requisitos con los roles y tareas identificados y descritos en la etapa del análisis de requisitos respectivamente.

3.2.3. *Diseño*

El diseño del sistema, consiste en abordar la manera de cómo representar la información (visualización, entradas, controles, etc.) al usuario. Se trata de un proceso en el que se traducirá toda la información obtenida hasta ahora de las etapas previas a una representación del software que se implementará.

El diseño todavía no proporciona detalles de implementación como la plataforma en la que mostrará la aplicación, etc. Simplemente se muestran los elementos que formarán parte de las interfaces de usuario.

El diseño de un sistema se puede abordar desde:

- Diseño de la estructura interna de los datos.
- Diseño arquitectónico.
- Diseño procedimiental de los componentes software.
- Diseño de la interfaz de usuario: establece la disposición de los mecanismos para la interacción hombre-máquina.

La etapa de diseño de las metodologías se corresponde con el estudio del dominio de la solución. Esta etapa trata de contestar el cómo integrar los requisitos no funcionales y cómo descubrir asuntos de implementación.

3.2.4. *Implementación*

En TOUCHÉ, la etapa de implementación del sistema consiste en generar la UI final (FUI) del usuario en base a los objetos abstractos de interacción definidos en la etapa de diseño. Es un proceso de reificación de cada uno de dichos componentes que concretizarán los elementos a emplear según los detalles de implementación y la plataforma objetivo, personalizaciones de usuarios, etc.

Esta estapa está muy influenciada por el marco de referencia Cameleon [Calvary et al., 2003] y por el lenguaje de descripción de interfaces de usuario

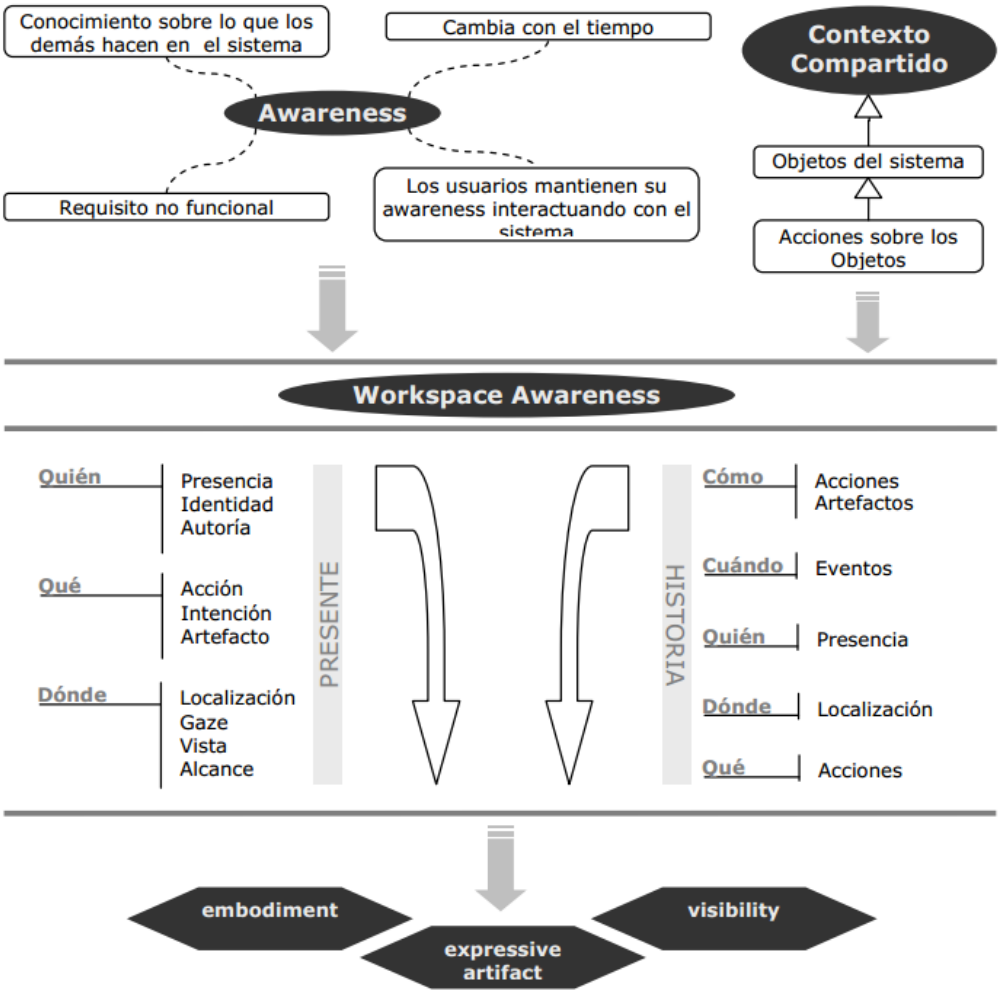


Figura 9. Elementos a tener en cuenta en los médos HCI para el diseñe la interfaz de usuario de los sistemas CSCW [Ruiz Penichet, 2007].

llamado UsiXML [Limbourg et al., 2005], el cual trataremos en el Capítulo 5.

En esta etapa se *traducen* los objetos abstractos de interacción a objetos concretos, dependiendo del contexto, de la modalidad y de otras especificaciones hechas en las fases anteriores.

Los objetos concretos de interacción seguirán una modalidad determinada. Por ejemplo, habrá componentes orientados a interfaces gráficas, otros orientados a interfaces vocales, etc.

3.3. CIAM

La metodología CIAM [Molina et al., 2009] (**C**ollaborative **I**nteractive **A**pplications **M**ethodology) está enfocada al desarrollo de sistemas Groupware.

El marco metodológico de CIAM está basado en la notación CIAN [Molina et al., 2008](**C**ollaborative **I**nteractive **A**pplications **N**otation), la cual permite definir los distintos modelos requeridos en las etapas de la metodología CIAM.

Los primeros estados de la metodología se refieren al modelado centrado en los grupos, llendo hacia un modelado más centrado en los procesos (colaborativos, cooperativos y de coordinación) en las etapas siguientes. Llendo cada vez más profundo en los niveles de abstracción se alcanza un modelado más centrado en los usuarios, en el cual las tareas interactivas son modeladas. Las primeras dos etapas del modelado permiten definir el contexto en el cual se crea el modelo interactivo, además de que sirve como punto de inicio para la última fase.

La información especificada en cada etapa sirve como base para el proceso de modelado hecho en las siguientes etapas, con lo cual dicha información es extendida, relacionada y especificada a mayor nivel de detalle en las siguientes etapas del proceso.

Las etapas y los objetivos de la metodología CIAM mostrados en la Figura

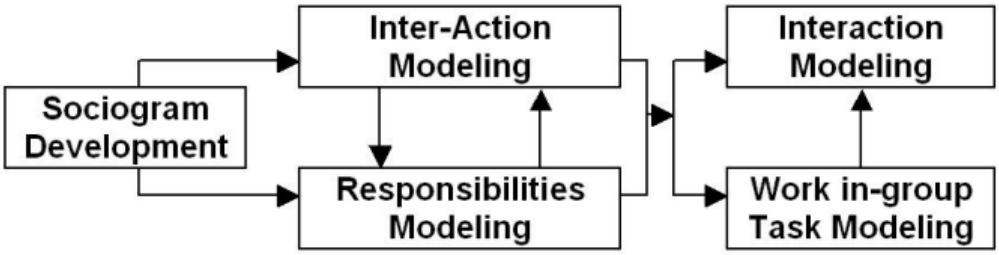


Figura 10. Etapas de la metodología CIAM [Molina et al., 2009].

10 son los siguientes:

1. **Desarrollo del sociograma**. En este estado se modela la estructura organizacional, así como las relaciones que existen entre sus miembros. Los miembros que forman la organización se encuentran en las siguientes categorías: *roles, actores, agentes de software*, o grupos de los anteriores, dando lugar a *grupos* o *equipos de trabajo*. Los elementos de estos diagramas pueden ser interconectados a través de tres relaciones básicas: *herencia, realización y asociación*.

2. **Modelado de las "***Inter-Action***"**: En esta fase se describen las principales tareas (o procesos) que definen el trabajo en grupo desarrollado en la organización definida en la etapa previa.

3. **Modelado de las responsabilidades**. En esta face, la atención es puesta sobre la perspectiva individual de cada rol en la organización, agregando a las responsabiledades compartidas aquellas que son exclusivas a cada rol. La información especificada en esta etapa es complementaria a la definida en las etapas previas, siendo necesaria para que los dos modelos sean coherentes.

4. **Modelado de las tareas de trabajo en grupo**. En esta fase se describen a mayor detalle las tareas grupales identificadas en las etapas previas. Se distinguen dos tipos de tareas, las cuales se modelan de forma distinta: *tareas cooperativas* y *tareas colaborativas*.

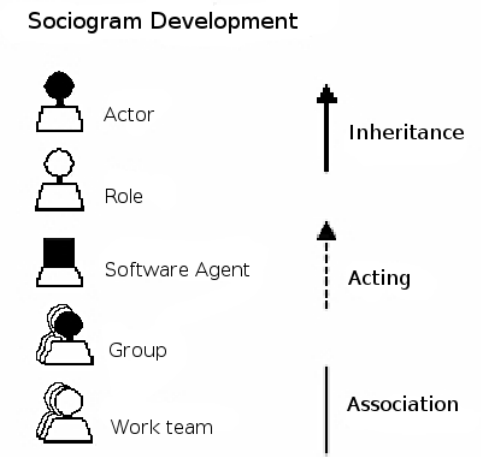


Figura 11. Elementos del sociograma en CIAM. Los elementos como Actor, Rol, Group, etc. pueden asociarse mediante las relaciones de Herencia, Asociación, etc.

5. **Modelado de la interacción**. En este último estado se modelan los aspectos de la aplicación púramente interactivos. Un modelo de interacción es creado para cada tarea de naturaleza individual.

3.3.1. *Etapa 1: Desarrollo del sociograma*

En esta etapa se define la estructura de la organización. El diagrama incluye las relaciones de herencia condicional que implica que un rol puede ser especializado bajo ciertas condiciones. La cardinalidad de cada rol es especificada a través de las relaciones entre los roles y los actores. Pueden asociarse varios roles que trabajan juntos en algunas actividades. Los elementos que componen el modelo del sociograma se muestran en la Figura 11

3.3.2. *Etapas 2 y 3: Modelado de Responsabilidades y de Inter-acción*

Los modelos elaborados en la etapa 2 y 3 se complementan entre sí, con lo cual pueden realizarse en orden indistinto.

Durante la elaboración de estos dos modelos hay una continua retroalimentación entre ellos, lo cual permite refinarlos a la vez.

Tasks \ Roles	Student	Director	Academic Committee	Examiners	Type
Draft Plan Writing	X	X			
Examining Board Proposal	X	X			
Request	X				
Academic Committee Valuation			X		
Suggest Changes			X		
DEP Development	X				
Post-DEP Procedure	X	X		X	
Examination	X			X	

Figura 12. Tabla de participación [Molina et al., 2008].

Task Types	Icon
Individual Tasks	
Cooperative Tasks	
Collaborative Tasks	

Figura 13. Tipos de tareas en CIAM [Molina et al., 2008].

En la etapa de modelado de Inter-Acción se crea la *tabla de participación* (Tabla 12), en la cual, usando una especificación de naturaleza textual, permite a los diseñadores tener una primera idea acerca de la división de trabajo a niveles altos de abstracción. Esta tabla está compuesta por las filas que representan las tareas del sistema. Las columnas representan los roles y cada celda significa que un rol participa en una tarea. La columna final identifica el tipo de tarea, ya sea individual, colaborativa o cooperativa, según se muestra en la Tabla 13.

El modelo de Responsabilidades expresa las responsabilidades que tiene un rol en particular sobre el conjunto de tareas en las que participa, según la Tabla de participación.

Responsibility	Task Type	Object in Domain Model	Pre-requirement	
			Task	Data
Draft Plan Writing		C/R/W: Draft Plan	INI	
Examining Board Proposal		R/W: Draft Plan	Draft Plan Writing	Draft Plan
Request		R/W: Draft Plan	- Draft Plan Writing - Examining Board Proposal	Draft Plan
DEP Development		C/R/W: DEP	Academic Committee Valuation	Draft Plan
Post-DEP Procedure		R/W: DEP	DEP Development	DEP
Examination		R/W: DEP	Post-DEP Procedure	DEP

Figura 14. Ejemplo de modelo de Responsabilidad para un rol ficticio llamado *Estudiante* [Molina et al., 2008].

Este modelo se expresa a través de una tabla (Tabla 14) cuyas filas son las tareas en las que participa un rol específico. Es decir, por cada rol va a haber una tabla o modelo de responsabilidades. La primera columna indica el nombre de la tarea a la que se refiere. La segunda columna indica el tipo de tarea. La tercera columna indica el objeto u objetos del dominio que son manipulados, así como el tipo de acceso requerido (R, Leer; W, Escribir; C, Crear; o combinaciones de éstos). La cuarta y quinta columna expresan los requisitos previos para poder realizar la tarea. Dichos requisitos consisten en la tarea que debe completarse para poder realizar la tarea a la que se refiere la columna, y los datos u objetos del modelo de datos que deben crearse.

Este modelo permite establecer las dependencias temporales o de orden de ejecución entre los procesos principales, así como las dependencias de datos a presentar.

Una vez que las responsabilidades y las tareas principales están definidas, se puede realizar el modelo de inter-acción. Este modelo permite definir la operación completa del proceso grupal, el cual puede ser cooperativo, colaborativo o mixto.

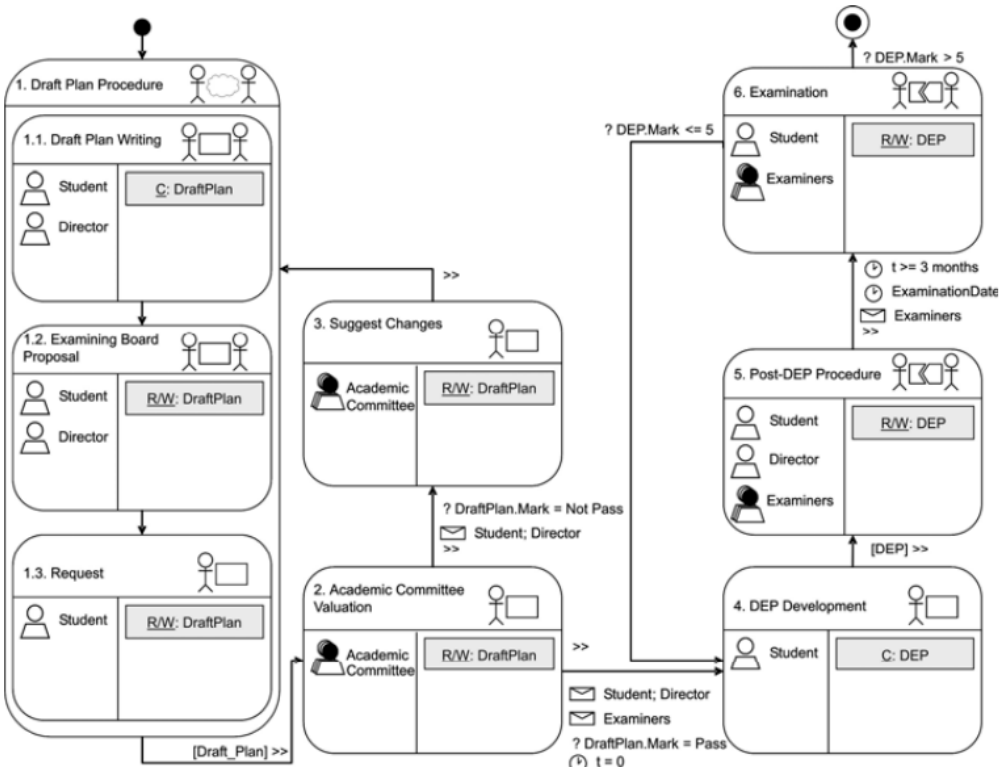


Figura 15. Diagrama de Inter-Acción de CIAM [Molina et al., 2008].

El modelo de inter-acción utiliza un diagrama que permite relacionar la información definida hasta el momento en las tablas de responsabiledad, tablas de participación y sociograma.

El diagrama del modelo de inter-acción (Figura 15) se representa mediante un grafo cuyos nodos son las actividades que componen el trabajo en grupo y cuyas aristas indican las relaciones entre estas actividades (de orden, dependencia, condición, notificación, paso de tiempo, etc).

Cada nodo incluye el nombre identificador de la tarea, el tipo de tarea, los roles participantes en su ejecución y los objetos manipulados junto con sus indicadores de acceso (C, Crear; R, Leer; W, Escribir).

Los operadores temporales disponibles en el modelo de inter-acción son los provistos por CTT [Paternò, 2004].

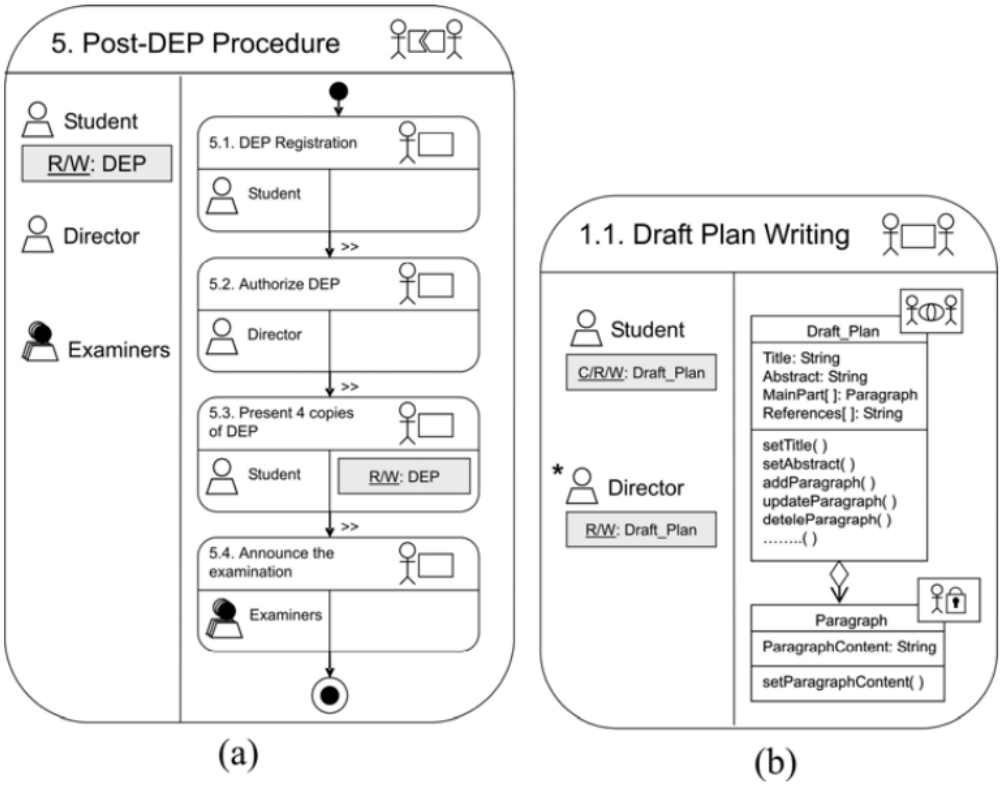


Figura 16. Ejemplo de modelado de dos tareas grupales en CIAM [Molina et al., 2008]: **a)** una tarea cooperativa y **b)** una tarea colaborativa.

3.3.3. *Etapa 4: Modelado de las Tareas de trabajo grupal*

En esta fase se profundiza en el detalle con el cual se especifican las tareas colaborativas y cooperativas, las cuales se diferencían en dos importantes aspectos [Dillenbourg et al., 1996]: las tareas cooperativas implican la división de la tarea cooperativa en tareas individuales llevadas a cabo por usuarios o actores con un rol particular, y las tareas colaborativas implican la manipulación de objetos o entidades que se comparten por los usuarios que desarrolla dicha tarea.

La Figura 16a muestra el modelado de una tarea cooperativa de ejemplo llamada *Post DEP Procedure*. A la izquierda de la tarea pueden verse los roles participantes en la tarea y los objetos manipulados por cada rol. En la derecha aparece la *gráfica de descomposición de responsabilidades*. Los nodos de la gráfica deben representar tareas individuales, las cuales involucran un sólo rol.

Icon	Definition
	Area of the shared context of collaborative visualization
	Area of the shared context of individual visualization
	Segment of the shared context of access of exclusive modification

Figura 17. Iconos para representar las características de visualización y acceso exclusivo al contexto compartido [Molina et al., 2008].

Para modelar las tareas colaborativas se requiere conocer los roles implicados en su ejecución y los objetos del modelo de datos que son manipulados en la misma de forma compartida. La Figura 16b muestra el modelado de la tarea colaborativa llamada *Draft Plan Writing*. A la izquierda de la tarea se muestran los roles implicados y los objetos manipulados junto con sus indicadores de acceso a ellos. A la derecha se muestran los objetos del modelo de datos que constituyen el contexto compartido. Para especificar el modelo compartido se utiliza la notación UML a la cual se le agregan algunos iconos para expresar características de visualización y para expresar los permisos de acceso al contexto compartido (Table 17).

3.3.4. *Etapa 5: Modelado de la Interacción*

En esta etapa los diseñadores se enfocan en especificar los diálogos que hay entre los usuarios individuales y las aplicaciones que median en el proceso colaborativo definido anteriormente. Entonces, para cada tarea individual es necesario crear un modelo de interacción. Para crear este modelo, los diseñadores deben identificar las tareas que son iniciadas por el usuario cuando el mismo interactua con la aplicación (tarea interactiva) y aquellas tareas que son ejecutadas enteramente por el ordenador (proceso interno o visualización de información, lo cual

en sí es una tarea de aplicación). Además, para modelar un dialogo de aplicación debe especificarse el orden temporal entre las tareas.

Para modelar esta interacción se utiliza la notación CTT [Paternò, 2004] (descrito más extensamente en la Sección 4.3.1), la cual está ampliamente diseminada en la comunidad de Interacción Persona-Ordenador.

El uso de CTT permite alcanzar mayores niveles de detalle en el modelo de interacción. Esto facilita la obtención del diseño final de la interfaz de usuario. En el caso particular de las tareas colaborativas, puede generarse directamente el árbol CTT que modela la interacción de este tipo de tareas.

Una ventaja de usar CTT es que ayuda a obtener la interfaz de usuario a partir del árbol de tareas CTT.

3.3.5. *Marco conceptual para modelar mecanismos de Awareness en CIAM*

Recientemente la metodología CIAM fue extendida [Gallego et al., 2011] para soportar el modelado de conceptos específicos del Awareness de Grupo [Gutwin and Greenberg, 2002] a través de un meta modelo desarrollado para incluir estos conceptos y a la vez proporcionar una base flexible para incluir otros conceptos relacionados con el Awareness.

A nivel de implementación cada uno de los conceptos del Awareness modelados puede traducirse en componentes de la interfaz de usuario gráfica (*Widget*), como se muestra en la Tabla 3. Dichos *widgets* están basados en los proporcionados por Groupkit [Roseman and Greenberg, 1996] y relacionados con el soporte del *Workspace Awareness* (Capítulo 3 Sección 4.2).

La idea final de la extensión es poder generar interfaces colaborativas a partir de los modelos de CIAM y los modelos de interacción, los cuales son enriquecidos con esta extensión para soportar elementos de Awareness.

Dichos esfuerzos en la extensión de la metodología CIAM muestran que existe una relación importante entre la interacción, la colaboración y el concepto

Widget	**Awareness**	**Descripción**
Vista de radar	Awareness de Sesión	Muestra el área en la cual los usuarios están trabajando y las actividades que están realizando.
Telepuntero	Awareness de Dominio	Cursor gráfico que indica la posición del puntero del Ratón (dispositivo periférico) de un usuario el cual está interactuando con el contexto compartido.
Propiedad de autoría	Awareness de Dominio	Muestra el usuario que creó cada objeto en el contexto comparido.
Lista de interacciones	Awareness de Sesión	Muestra textualmente las interacciones que toman lugar durante la sesión de trabajo.
Barra de desplazamiento multi usuario	Awareness de Dominio	Muestra la localización de cada usuario en el área de trabajo de acuerdo a la posición de su barra de desplazamiento.
Anotación	Awareness de Session	Nota que un usuario puede publicar a fin de compartir mensajes asíncronamente.
Panel de sesión	Awareness de Sesión	Muestra los actores que se encuentra en el sistema, incluyendo información adicional de ellos.
Selección remota	Awareness de Dominio	Muetra objetos en el contexto compartido que han sido seleccionados por otros usuarios.

Tabla 3. *Widgets* para el soporte del Awareness en CIAM.

llamado Awareness.

4. Componentes generales

En esta sección describimos algunas técnicas utilizadas de forma general por las metodologías descritas en la Sección 3 y que también se utilizan en los capítulos posteriores para representar los ejemplos de aplicación.

Englobamos las técnicas descritas de acuerdo a su finalidad en el proceso de desarrollo, como puede ser el análisis de requisitos o el modelado de la estructura de tareas del sistema.

Las diferentes técnicas o componentes que presentamos a continuación están divididos en las siguientes categorías: Para el análisis de requisitos, para la definición de la estructura del dominio, para la definición de las tareas y para la definición de los aspectos interactivos.

4.1. Análisis de requisitos

El análisis de requisitos se refiere a la fase en la que se adquiere, se documenta y tal vez se modela la información del negocio o dominio de la aplicación. Esta etapa normalmente es la primera durante el proceso iterativo de desarrollo, ya que sin conocer el dominio y los requerimientos del sistema poco se puede hacer.

Dependiendo de la metodología la fase de obtención y análisis de requisitos podrá enfocarse a un aspecto del sistema o en varios a la vez. Sin embargo, lo importante en esta etapa es conocer de forma general el sistema y lo que se necesita de él a fin de conseguir los objetivos que cada usuario tiene.

En esta sección describimos los Casos de Uso de UML utilizados para elicitar requisitos de actividades o tareas realizadas por el usuario. Otra forma de representar requisitos son las plantillas de requisitos, las cuales se asemejan a formularios predefinidos que permiten especificar requerimientos y/u objetivos que debe cumplir el sistema. Un ejemplo de estas plantillas son las presentadas por [Penichet et al., 2009a].

Los Casos de Uso de UML [Group, 2001] son una técnica importante para el

análisis de requerimientos. Un caso de uso es una serie de pasos que definen la interacción entre un rol o (Actor en UML) y el sistema, a fin de consiguir un objetivo determinado. Son una buena técnica para comenzar a extraer requisitos, ya que permite comenzar a identificar los roles del sistema, las tareas de más alto nivel y los objetivos en cada una de ellas.

4.2. Representación del Dominio

Una forma de representar el dominio de un sistema es representar las entidades que lo componen, las relaciones entre dichas entidades y su comportamiento o características singulares. Para este propósito se pueden utilizar los Diagramas de Clases UML [Group, 2001], los cuales son bastante populares dada su sencillez y capacidad de representación.

El diagrama de clases en UML permite representar la estructura lógica de alguna entidad en el sistema y su relación con otras entidades.

4.3. Tareas

De forma más general, los modelos de tareas pueden usarse para muchos propósitos [Paternò, 1999]:

- *Mejoran el entendimiento del dominio de aplicación.* La actividad del modelado ayuda a los diseñadores a clarificar cuáles tareas deberían soportarse, su relaciones mutuas, sus restricciones, etc.
- *Guarda el resultado de una discusión interdisciplinaria.* Ya que estos modelos son la descripción lógica de las actividades a soportar y, por lo tanto, deberían considerar todos los puntos de vista de las partes interesadas (diseñadores, desarrolladores, usuarios finales, expertos en el dominio, clientes, etc.).
- *Soportar un diseño efectivo.* Si se crea una correspondencia directa entre las tareas y las interfaces de usuario, entonces los usuarios pueden alcanzar

sus objetivos efectiva y satisfactoriamente.

- *Documentación.* Los modelos de tareas pueden ser útiles para proveer documentación acerca de como usar el sistema de acuerdo a las tareas soportadas.

Dado que en este trabajo hacemos amplio uso de la notación ConcurTaskTrees [Paternò, 1999], ocuparemos esta sección para describirla de forma general, a fin de que sirva como guía para enteder los ejemplos en capítulos posteriores.

4.3.1. *ConcurTaskTrees CTT*

ConcurTaskTrees es una notación para la especificación del modelo de tareas, la cual ha sido desarrollada para diseñar aplicaciones interactivas. Su principal propósito es el de ser una notación fácil de usar que pueda soportar el diseño de apliaciones industriales reales, lo cual usualmente significa aplicaciones de medianas o grandes dimensiones.

Las principales características de CTT son:

- Se enfoca en actividades, lo cual permite a lo diseñadores el concentrarse en las actividades que los usuarios pretenden realizar, lo cual es el aspecto más importante cuando se están diseñando aplicaciones interactivas que abarcan a los usuarios y a los aspectos relativos al sistema, evitando detalles de la implentación a bajo nivel que a nivel de diseño sólo oscurecen las decisiones que deben tomarse.
- Tiene una estructura jerárquica, la cual resulta algo muy intuitivo. De hecho, a menudo las personas tienden a descomponer un problema a resolver en problemas más pequeños, manteniendo las relaciones entre todas las partes de la solución. La estructura jerárquica de la especificación de CTT tiene dos ventajas: provee un amplio rango de granularidad permitiendo la reutilización de grandes o pequeñas estructuras de tareas, y permite estructuras de tareas reusables definibles a un nivel semántico bajo o alto.

- Proporciona una sintaxis gráfica, la cual a menudo (aunque no siempre) es más fácil de interpretar. En el caso de CTT, la sintaxis gráfica refleja la estructura lógica en forma parecida a la de un árbol.

- Proporciona un grupo de operadores temporales los cuales se utilizan para definir relaciones entre las tareas. El hacer que los analistas usen estos operadores cambia substancialmente las prácticas normales. La razón para esta innovación es que despues del análisis informal de las tareas, queremos que los diseñadores expresen claramente las relaciones lógicas y temporales. Esto se debe a que dicho orden debe tomarse en cuenta en la implementación de la interfaz de usuario, la cual permite a los usuarios llevar a cabo las tareas queden estar activadas desde el punto de vista semántico.

- Permite representar los objetos y atributos de la tarea. Una vez que la tarea es identificada, es importante indicar los objetos que deben ser manipulados en ella, a fin de poder realizarla. Se consideran dos tipos de objetos en CTT: los objetos de la interfaz de usuario y los objetos del dominio de la aplicación. Varios objetos de la interfaz de usuario pueden asociarse a objetos del dominio.

La notación CTT ha mostrado dos resultados positivos: **(1)** una notación expresiva y flexible capaz de representar actividades interactivas y concurrentes, además con la posibilidad de soportar cooperación entre múltiples usuarios y posibles interrupciones, **(2)** una representación entendible y compacta, cuyo aspecto clave ha sido la habilidad de proveer muchar información de una manera intuitiva sin requerir un esfuerzo excesivo de los usuarios de la notación, aún sin tener un fondo en ciencias de la computación o en informática.

Es posible especificar directamente para cada tarea simple un número de atributos junto con información relacionada. Estos atributos e información se clasifican en tres secciones:

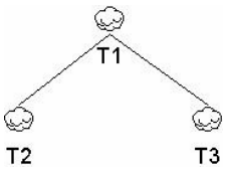
- *Información general*, la cual incluye el nombre e identificador de la tarea,

su categoría y tipo, frecuencia de uso, algunas anotaciones informales, indicadores de precondiciones posibles y si es una tarea interactiva, opcional y de conexión.

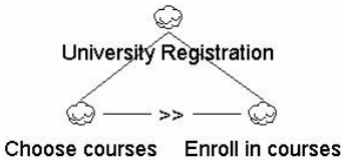
- *Objetos.* Para cada tarea es posible indicar objetos (con nombre y clase) que van a ser manipulados durante la ejecución de la tarea. Los objetos pueden ser de la interfaz de usuario o del dominio de la aplicación. También es posible indicar los derechos de acceso que pueden tener los usuarios sobre los objetos cuando ejecutan dicha tarea. Es posible indicar también la plataforma en la cual se considera el uso del objeto.

- *Tiempo de ejecución de la tarea.* Varios valores sobre el tiempo de ejecución de la tarea pueden indicarse, tales como mínimo y máximo y promedio.

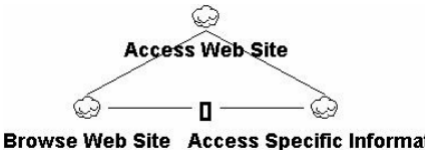
- **Jerarquía**. Las tareas al mismo nivel representan diferentes opciones o diferentes tareas al mismo nivel de abstracción al cual deben ser ejecutadas. Leer los niveles como "*a fin de ejecutar T1, necesito ejecutar T2 y T3*", o "*a fin de ejecutar T1, necesito ejecutar T2 o T3.*".



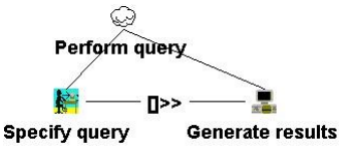
- **Habilitación**. Especifica que una segunda tarea no puede empezar hasta que finalice la primera tarea. Por ejemplo, Yo no puedo inscribirme a la Universidad si antes no he escogido los cursos que quiero tomar.



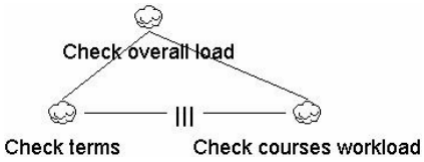
- **Opción**. Especifica dos tareas habilitadas (relación binaria, pero encadenadas pueden ser más). Una vez que una de ellas inicia la otra ya no está habilitada.



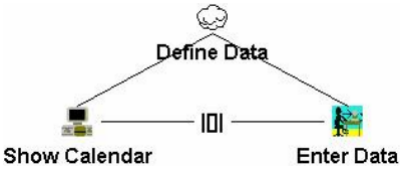
- **Habilitación con paso de información**. Especifica que la segunda tarea no puede ejecutarse sin antes haber ejecutado la primera, y que la primera tarea una vez finalizada debe enviar información a la segunda tarea para que ésta pueda comenzar.



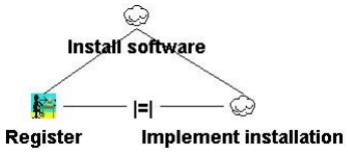
- **Tareas concurrentes**. Las tareas pueden ejecutarse en cualquier orden o al mismo tiempo, incluyendo la posibilidad de empezar una tarea antes de que la otra sea completada.



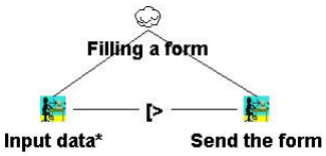
- **Tareas concurrentes comunicativas**. Especifica tareas que intercambian información mientras son ejecutadas concurrentemente.



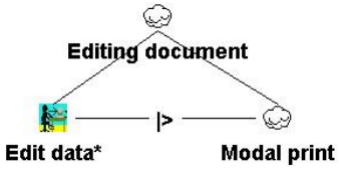
- **Tareas independientes**. Las taraes pueden ejecutarse en cualquier orden, pero cuando una inicia la otra no pueden empezar hasta que la iniciada haya finalizado.



- **Deshabilitación**. La primera tarea (usualmente una tarea interactiva) es completamente interrumpida por la segunda tarea.



- **Suspeción - Reanudación**. La primera tarea puede ser interrumpida por la segunda. Cuando la segunda termina entonces la primera puede ser reactivada al estado en el cual se encontraba cuando fue interrumpida.



En CTT existen cuatro tipos de tareas:

- Tarea abstracta: Son acciones complejas que no son parte de una única categoría.

- Tarea de interacción: Es una tarea en la que el usuario interactúa con el sistema.

- Tarea del sistema: Es una tarea que se ejecuta completamente por el sistema.

- Tarea del usuario: Son tareas ejecutadas por el usuario pero sin interactuar con el sistema.

- Tarea colaborativa: Son tareas compuestas por subtareas donde hay varios usuarios para realizar las tareas básicas.

Las tareas en CTT pueden estar compuestas por otras subtareas de diferente tipo (Tarea Abstracta). Esto permite agrupar un conjunto de tareas en un nivel de abstracción superior y así poder representar las tareas del sistema a distintos niveles.

El uso de CTT permite obtener la interfaz de usuario de forma parcial o total. La notación CTT ha sido usada como punto de partida para varias propuestas para obtener o generar la interfaz de usuario de forma semi automática [Coninx et al., 2003], [Mori et al., 2004].

La notación CTT también será utilizada en algunos ejemplos de este trabajo debido a su popularidad, sencillez y a su aceptación general en la comunidad del desarrollo de software.

5. Comparativa de las metodologías y el soporte de la colaboración

En las metodologías revisadas anteriormente encontramos varios componentes para el soporte de la colaboración: Requerimientos de "*conciencia*" en las plantillas de requerimientos del sistema, Definición de roles, definición de tareas grupales, definición del contexto compartido y definición de la conciencia de grupo. A continuación destacamos las técnicas utilizadas para el soporte del trabajo grupal y en especial las relacionadas con el aspecto conocido como conciencia grupal o Awareness.

Las plantillas para requerimientos de sistemas Groupware descritas en [Penichet et al., 2009a] incluyen un apartado para representar los requerimientos de conciencia, los cuales son utilizados en TOUCHÉ para describir la información que el usuario debe tener disponible para tener una conciencia de grupo según los trabajos de Gutwin sobre la conciencia del espacio de trabajo compartido (Capítulo 3 Sección 4.2).

La definición de los roles de la organización es prácticamente igual a la definición de la estructura organizacional y de grupos. Se puede realizar a través de Diagramas de Casos de Uso o de Clases en UML, así como con las notaciones propuestas por las metodologías TOUCHÉ (Sección 3.2) , AMENITIES (Sección 3.1) o CIAM (Sección 3.3).

La definición de tareas grupales depende de si la tarea es colaborativa (contexto compartido) o cooperativa (distribución de trabajo en tareas individuales). La metodología CIAM proporciona diagramas específicos para cada tipo de tarea. En la notación CTT sólo pueden representarse tareas colaborativas, de las cuales pueden hacerse varios árboles dependiendo de cada rol que participe en la tarea.

CIAM también proporciona técnicas para representar el contexto compartido el cual puede ser información, objetos o artefactos que pueden proporcionarse a los usuarios de distintos roles si éstos trabajan en una tarea colaborativa. AMENITIES utiliza para este propósito los Artefactos y los Objetos de Información.

En cuestión del aspecto de la conciencia de grupo, CIAM permite al definición de algunos aspectos de la conciencia de grupo a nivel de diseño. Por otra parte, TOUCHÉ plantea las necesidades de conciencia de grupo desde los requerimientos y las soporta a nivel de diseño mediante interfaces de usuario especializadas.

Aparte de las técnicas utilizadas por las metodologías revisadas, existen varias técnicas y herramientas para el soporte de la conciencia grupal a nivel de diseño, algunas de las cuales presentaremos en el Capítulo 3 Sección 5.

El Awareness en las técnicas de modelado			
Etapa	**AMENITIES**	**TOUCHÉ**	**CIAM**
Requisitos		Plantillas de requerimientos.	
Objetivos		Plantillas de objetivos.	
Sociograma			
Roles			
Tareas	Artefactos y objetos de información.		
Tareas colaborativas			Contexto compartido.
Tareas cooperativas			
Interacción			
Interfaz de usuario	Artefactos de comunicación.	Widgets especiales.	Widgets especiales.
Trazabilidad			

Tabla 4. La inclusión del Awareness en las técnicas de modelado de las metodologías analizadas.

Es notorio que de una u otra manera, los aspectos relacionados con la colaboración/cooperación entre usuarios está presentes de principio a fin del proceso de desarrollo de un Groupware. Si el Awareness es parte del proceso interactivo, también influirá en todas las fases del desarrollo, o por lo menos en la mayoría, especialmente si nos enfocamos en los sistemas Groupware.

La Tabla 4 muestra el resumen del soporte del Awareness en las metodologías analizadas. Alguna lo toma en cuenta en los requisitos. Otra en el diseño y prácticamente todas en las interfaces de usuario. Ninguna metodología revisada define alguna técnica para mantener la trazabilidad del soporte del Awareness a lo largo del proceso de desarrollo, aun siguiendo un proceso dirigido por modelos.

6. Conclusiones

Después de analizar las características generales de los sistemas Groupware y revisar tres metodologías de desarrollo para el Groupware encontramos varias

similitudes en el aspecto conceptual, aunque técnica y logísticamente tengan diferencias considerables.

La descripción de los usuarios (roles), de la organización y de los grupos de trabajo tiene gran importancia dado que de esta manera se pueden dislumbrar las actividades que cada uno realiza en individual y las que se realizan en grupo.

La representación de las tareas también cambia, ya que puede haber tareas en las que participa más de un usuario y puede haber otras que se completan solo cuando varios usuarios finalizan tareas individuales relacionadas con dicha tarea.

Las tres metodologías admiten la inclusión de algunas formas de conciencia grupal como parte de la metodología, especialmente en la parte de los requerimientos y en las interfaces de usuarios.

El soporte de la llamada conciencia grupal o Awareness se centra en el espacio compartido y en proveer información sobre el mismo a los demás usuarios. A través de la comparativa nos percatamos de que no se encontraron formas para especificar y restringir la información de Awareness manejada. Además, no todos los componentes de dichas metodologías manejan el concepto de Awareness, por lo que entre las distintas etapas del desarrollo el soporte del Awareness se pierde o desliga de los mecanismos de trazabilidad elaborados en algunas metodologías y técnicas. Además, dichas metodologías buscan soportar el Awareness por lo menos a nivel diseño y en las interfaces de usuario, con lo cual se confirma que el Awareness está presente en todas las metodologías revisadas.

El enfoque sobre el Awareness en las metodologías es que está relacionado fuertemente con la colaboración, aunque su relación no es tan clara con la cooperación y con las tareas individuales.

La comunidad Groupware en general parece estereotipar el Awareness como exclusivo de los procesos colaborativos, cosa que de antemano tenemos como hipótesis que es falso. Sin embargo, esta afirmación requiere un estudio más

profundo del Awareness en otras áreas, de tal manera que podamos entender sus características y usos más apropiados a fin de mejorar el desarrollo de los sistemas Groupware en este aspecto.

No solo las metodologías orientadas al Groupware pueden usarse para desarrollar Groupware. Otras metodologías se usan también para eso, aunque teniendo que lidiar con aspectos únicos del Groupware (colaboración y cooperación) sin ayudas por parte de la metodología (Desarrollo tradicional, desarrollo de sistemas con nuevos paradigmas, etc.).

Partiendo de este entorno procedemos a elaborar un estado del arte sobre el Awareness para encontrar otros puntos de vista que nos permitan confirmar si el soporte del Awareness encontrado en algunas metodologías especializadas en el Groupware es suficiente para soportar el Awareness ocupado en sistemas ya desarrollados.

CAPITULO 3

Conciencia situacional y grupal: Awareness

1. Introducción

La necesidad de soportar tareas colaborativas a través del ordenador generó la aparición de un tipo de sistemas software específicos para dicha labor: los sistemas Groupware.

Como vimos en el Capítulo 2, los sistemas Groupware buscan proveer un entorno adecuado para soportar la realización de tareas colaborativas y/o cooperativas a sus usuarios. Estos sistemas deben soportar importantes flujos de interacción y por consiguiente, resolver todas las dificultades relacionadas con ello. Su principal dificultad es el soporte de la colaboración.

Como veremos en este capítulo, diferentes estudios en el área de los sistemas Groupware han encontrado una clara relación entre un eficiente soporte de la colaboración y el de un aspecto de la interacción crucial para cualquier ser vivo: la conciencia de grupo o *Awareness* (término en inglés).

En este capítulo mostramos un ańalisis de los aspectos más relevantes sobre el Awareness y su relación con el software, ya que el objetivo de este trabajo es mejorar el soporte de este aspecto complejo, pero inherente a cualquier ser vivo, y del cual dependen todos los procesos interactivos y colaborativos.

Aunque el soporte del Awareness en el software es un tema que ha sido abordado desde diferentes perspectivas, siguen faltando técnicas y recursos para su

correcto soporte a lo largo y ancho del desarrollo de un software interactivo.

En las siguientes secciones mostramos un panorama general del Awareness a través de los estudios realizados el área del Groupware centrándonos en su soporte y en las técnicas y recursos existentes.

A través de este estudio podemos constatar que todavía existen varias dificultades en el soporte del Awareness las cuales todavía impiden su correcta definición, manipulación y soporte, especialmente si se quiere integrar en procesos de desarrollo badasos en modelos (MDD)[1].

A continuación se describe la estructura de este capítulo. En la Sección 2 se presentan varias definiciones del término Awareness en el contexto de los sistemas software. En la Sección 3 se describe el papel del Awareness en la interacción y colaboración, así como su efecto en los sistemas Groupware. En la Sección 4 mostramos un recopilatorio de los tipos de Awareness más comúnes en la literatura científica sobre el Awareness, describiendo sus características y entornos de aplicación.

Después de las secciones enfocadas a la descripción del Awareness, continuamos con la Sección 5, en la cual se presentan los diferentes enfoques para el soporte del Awareness en las distintas fases del desarrollo del software. La Sección 6 presenta un estudio como parte de este trabajo de las interfaces y técnicas de interacción usadas para proporcionar Awareness así como sus características principales.

La Sección 7 muestra las dificultades que derivan del soporte del Awareness. También se muestran las dificultades que atraviesan los usuarios a la hora de recibir, procesar y utilizar el Awareness que les proporciona el sistema sobre su entorno y sobre los otros usuarios. Estan dificultades resaltan la necesidad de modelos conceptuales adecuados para el desarrollo del software, así como su integración en las metodologías utilizadas para desarrollar estos sistemas.

[1] `http://en.wikipedia.org/wiki/Model-driven_development`

Para finalizar y después de resumir algunos aspectos del soporte del Awareness en la Sección 8, concluimos el estudio del Awareness en la Sección 9, en la cual se muestran los aspectos más importantes del Awareness, su soporte actual y las dificultades tanto para desarrolladores como para usuarios.

2. Definiciones

La RAE[2] (Real Academia Española) proporciona varias definiciones para el término *conciencia*:

1. Propiedad del espíritu humano de reconocerse en sus atributos y en todas las modificaciones que en sí mismo experimenta.
2. Conocimiento interior del bien y del mal.
3. Conocimiento reflexivo de las cosas.
4. Actividad mental a la que solo puede tener acceso el propio sujeto.
5. Acto psicológico por el que un sujeto se percibe a sí mismo en el mundo.

Las definiciones de la RAE no son muy fáciles de interpretar y ciertamente no ayudan mucho a comprender lo que es la *conciencia* y cómo soportarla en los sistemas software. Lo que sí se puede entender a través de estas definiciones es que la conciencia es un proceso o producto mental.

El diccionario de Oxford[3] (lengua inglesa) define *Awareness* como "*knowledge or perception of a situation or fact*", lo cual, traducido al español significa ("*Conocimiento o percepción de una situación o hecho.*").

El término **Awareness** o **conciencia de grupo** es más familiar para la comunidad del Groupware, ya sea por su definición más sencilla y genérica o por

[2] `http://www.rae.es`
[3] `http://oxforddictionaries.com`

los trabajos sobre el Awareness en idioma inglés. En este trabajo utilizaremos principalmente el término inglés por estas mismas razones.

El término Awareness ha sido definido de diversas formas, por diferentes autores pero siempre de acuerdo a necesidades específicas y en contextos muy determinados.

Una de las dificultades para el soporte del Awareness es esa gran cantidad de definiciones y terminologías ligadas a este concepto [Schmidt, 2002]. Es decir, cada cual ve al Awareness de forma distinta, aún en la misma área de investigación. Sin embargo, a través de estas definiciones podemos encontrar los puntos en común que hay en todas ellas y de esta forma entender como los desarrolladores e investigadores entienden el concepto de Awareness.

Varias definiciones están ligadas a formas o tipos de Awareness específicos y comúnmente encontrados en la literatura. Estos tipos de Awareness serán presentados y detallados en la Sección 4.

A continuación se muestra un listado de algunas de las definiciones del Awareness encontradas en la literatura científica referente a los sistemas software:

Awareness: El entendimiento de las actividades de los otros, las cuales proporciona el contexto para nuestras propias actividades [Dourish and Bellotti, 1992].

Awareness: El conocimiento de lo que está pasando [Endsley, 1995].

Awareness de Situación: La percepción de los elementos del entorno en un volumen de espacio y tiempo, la comprensión de su significado y la proyección de su estado en un futuro cercano [Endsley, 1995].

Awareness de concepto: El entendimiento que tienen los participantes de cómo sus tareas serán completadas [Gutwin et al., 1995].

Awareness social: El entendimiento que tienen los participantes acerca de las conexiones sociales dentro de sus grupos [Gutwin et al., 1995].

Awareness de tarea: El entendimiento de los participantes acerca de cómo sus tareas van a ser completadas [Gutwin et al., 1995].

Awareness de espacio de trabajo: El conocimiento actualizado de las interacciones de los otros participantes con el espacio de trabajo compartido [Gutwin et al., 1995].

Awareness de la estructura de grupo: El conocimiento acerca de cosas como los roles de las personas y sus responsabilidades, sus posiciones en un equipo, su estado y los procesos de grupo [Gutwin et al., 1996a].

Awareness informal: La sensación general de quienes están alrededor y de lo que otros están haciendo [Gutwin et al., 1996a].

Awareness periférico: La ubicación de las personas en el contexto global [Gutwin et al., 1996b].

Awareness conversacional: Quién está comunicándose con quién [Vertegaal et al., 1997].

Awareness de espacio de trabajo: Quién está trabajando en qué [Vertegaal et al., 1997].

Awareness social: Información acerca de la presencia y actividades de las personas en el entorno compartido [Prinz, 2002].

Awareness orientado a la tarea: Awareness enfocado a las actividades ejecutadas para completar una tarea compartida [Prinz, 2002].

Awareness de tarea: El entendimiento de la tarea colaborativa [Kim and Kim, 2007].

Awareness de los miembros: El conocimiento de la información actual de los miembros del equipo, como son sus roles en el proyecto, sus perfiles de interés y sus conocimientos. [Kim and Kim, 2007].

Awareness de presencia: El conocimiento de la presencia en el sistema de los otros miembros del equipo. [Kim and Kim, 2007]

Awareness de planificación: El conocimiento de la planificación temporal (horarios) de los otros miembros del equipo, relacionado con el proyecto o actividad [Kim and Kim, 2007].

Awareness de actividad: El conocimiento de las actividades de los otros [Kim and Kim, 2007].

Awareness de actividad: El Awareness del trabajo del proyecto, el cual soporta el trabajo en equipo durante las tareas complejas [Carroll et al., 2003].

Awareness del ritmo: El modelo cognitivo de patrones temporales recurrentes de algún evento particular. Se refiere al conocimiento actualizado de los patrones temporales formados por la recurrencia de un evento, como puede ser la disponibilidad de una persona, el cambio de estado de un dispositivo, etc. [Begole and Tang, 2007].

Awareness del conocimiento: El Awareness del uso del conocimento [Ogata et al., 1996].

Awareness del conocimiento compartido: El Awareness del conocimiento compartido y de su construcción [Collazos et al., 2003].

A partir de todas las definiciones podemos obtener algunas conclusiones generales con respecto al Awareness:

- El Awareness es un estado mental.
- Dicho estado mental se refiere al conocimiento actualizado de un ente, ya sea el entorno, una situación, un objeto u objetos, una idea o pensamiento, etc.
- El Awareness está ligado al tiempo y al espacio, aunque su desarrollo es en el tiempo presente.

- No es lo mismo *Awareness* y *conocimiento*, ya que el conocimiento se refiere a algo que ya está asimilado en la memoria de largo plazo, en cambio, el Awareness se refiere al conocimiento actualizado, el cual se encuentra todavía en la memoria de trabajo, lo cual apunta al tiempo presente.

- El Awareness está ligado a entes con características o atributos cambiantes.

- El Awareness está ligado a la percepción. No se puede tener Awareness sin tener también la capacidad de adquirir datos del exterior. En los seres humanos, los sentidos nos permiten percibir datos del exterior. Sin esos sentidos no es posible tener Awareness del exterior, ya que el exterior cambia constantemente.

A fin de comenzar nuestros trabajos sobre una base téorica fundamentada y suficientemente genérica para incluir todos los tipos de Awareness, hemos elegido como base para este trabajo la definición de Mica R. Endsley sobre el Awareness de Situación. Esta definición será la base para toda nuestra propuesta.

3. Papel del Awareness en la interacción y en la colaboración

El Awareness como estado mental viene implícito en los seres vivos, tengan o no conciencia de sí mismos. Este estado les permite interactuar con el entorno en tiempo presente.

Esto quiere decir que el Awareness es necesario para la interacción de una entidad con su entorno próximo y perceptible.

La interacción siempre ocurre en tiempo real, es decir, en el presente. Un individuo puede basarse en interacciones anteriores para realizar una interacción en el presente. Sin embargo, si el entorno cambia, el conocimiento generado por las interacciones anteriores puede no servir para predecir el estado presente, ya que el entorno puede ser diferente en algún aspecto.

Interactuar con el entorno sin tener Awareness de su estado puede impedir la realización efectiva de las actividades que se requieren. Por ejemplo, si una persona quiere tomar un vaso de la mesa pero no lo puede ver, probablemente intente localizarlo mediante el tacto. Si la persona no posee tacto, entonces nunca va a saber si ya lo tiene en la mano o si lo ha tocado. En términos prácticos, le va a ser casi imposible tomar el vaso.

Esta claro que la primera necesidad para poder interactuar con el entorno es el Awareness. La primera necesidad del Awareness es la percepción.

La percepción implica la capacidad sensorial para adquirir información del exterior. Sin la percepción, la construcción del estado mental asociado al Awareness no es posible.

La capacidad mental de entender, formar patrones, inferir, le permite tomar informarción del exterior y utilizarla como materia prima para sus procesos mentales.

La definición del Awareness de Situación incluye también el entendimiento del entorno y la proyección de su estado en un futuro cercano. Estas capacidades derivan de los procesos mentales de los que es capaz el ser humano, algunos animales y los agentes artificiales (utilizando técnicas de inteligencia artificial para simular estos procesos).

El papel del Awareness en la colaboración se deriva de que el proceso de colaboración requiere una mayor cantidad de interacción, especialmente con otros individuos. Ya sea para comunicarse entre sí o para realizar una tarea conjunta, las necesidades de interacción crecen cuando se realiza una tarea colaborativa o cuando coopera con otros inviduos.

En el caso de los seres humanos, éstos utilizan diversas formas de comunicación basadas en los sentidos y en habilidades acordes a cada uno de ellos: la voz para la audición, la escritura para la visión, el alfabeto Braille para el tac-

to. La percepción y la generación de señales soportan todos los mecanismos de comunicación.

El Awareness no solo se limita a cuestiones como *¿Qué hora es?* o *¿Cuánto tiempo llevo caminando?*, sino que también abarca cuestiones como *¿Quién está hablando?* o *¿Qué está diciéndome?*.

El soporte de la colaboración en el software se encontró con la necesidad de soportar también el Awareness en sus diversas formas, similares en algunos casos pero nunca iguales, ya que cada software y cada dominio presentan necesidades diferentes.

Las cuestiones que aparecen entonces son ¿qué tipo de Awarenss se necesita?, ¿cómo se define?, ¿Cómo se soporta? y ¿cómo se mantiene?

En la siguiente Sección presentamos un estado del arte sobre los tipos de Awareness más comunes en la lituratura científica, así como las diversas técnicas y herramientas utilizadas para su soporte.

4. Tipos de Awareness

Dada la diversidad de interpretaciones del Awareness, se pueden encontrar muchos tipos de Awareness para diferentes entornos, dominios y aplicaciones. No obstante, gran parte de ellos comparten objetivos comunes y han sido estudiados y catalogados de acuerdo a ello.

A continuación presentamos un análisis de los distintos y más comunes tipos de Awareness encontrados en la literatura científica referente a los sistemas Groupware. La descripción de cada tipo de Awareness constará de los siguientes elementos: Introducción, entornos de uso, herramientas de soporte y notas adicionales.

4.1. Situation Awareness

El Awareness de Situación o *Situation Awareness* (SA) ha sido descrito ampliamente por Mica R. Endsley en su Teoría del *Situation Awareness* para sistemas dinámicos y complejos [Endsley, 1995]. En este trabajo se analiza todo lo que respecta al *Situation Awareness*, desde los aspectos más básicos basados en las teorías de la psique humana, hasta su efecto determinante en el proceso de toma de decisiones en entornos dinámicos, complejos y en varios casos, de supervivencia. La principal motivación de este estudio fue el diseño de vehículos militares y la relación existente entre la información actualizada del entorno necesaria para los pilotos y la capacidad de procesarla para cumplir sus objetivos.

Ya que la supervivencia de los pilotos depende de sus capacidades para tomar decisiones de forma rápida y efectiva en entornos cambiantes, la implicación del *Situation Awareness* como estado de conocimiento para este propósito es innegable.

Podemos decir que el *Situation Awareness* es el Awareness más genérico de todos los descritos. El Awareness de Situación es la base de todos los tipos de Awareness existentes, ya que el término "*Situation*" puede utilizarse para describir cualquier cosa, desde la presencia, la localización, hasta los conceptos más abstractos como el contexto, el conocimiento compartido, la creatividad, etc.

Descripción

El Awareness de Situación es *"la percepción de los elementos del entorno en un volumen de tiempo y espacio, la comprensión de su significado y la proyección de su estado en un futuro cercano"* [Endsley, 1995].

Cada situación de importancia para la toma de decisiones es definida en base a los elementos del entorno, los cuales pueden ser cualquier atributo o característica perceptible del entorno, dependiendo de los objetivos del humano o agente que los percibe y utiliza.

Para la formación del Awareness, la percepción está siempre ligada al tiempo, por lo cual el Awareness representa un estado de conocimiento alimentado por la constante percepción de datos del entorno. Además, a través de procesos de inferencia, deducción o lógica, el observador puede formar una comprensión de los valores de los elementos en su conjunto, aumentando con ello su nivel de conciencia o Awareness. Este nivel de comprensión no va más allá de una sola observación o conjunto de valores percibidos. Es decir, si en una observación se obtiene un conjunto de valores de determinados elementos del entorno, el nivel de comprensión le permite al observador crear nuevo conocimiento a partir de los valores percibidos.

Como último nivel del Awareness como estado de conciencia encontramos la proyección de los valores de los elementos en un futuro cercano. Esto responde a la pregunta *¿Qué valores tendrán los elementos en el tiempo X (el cual está en el futuro)?*

Este tipo de Awareness fue el primero formalmente descrito mediante un marco teórico completo y generalizado para poder ser aplicado a cualquier situación.

Aunque al principio no se le relacionó con otros tipos de Awareness de importancia, es la base estructural y conceptual de todos ellos. Por tanto, todos los Tipos de Awareness pueden construirse como una especialización del *Situation Awareness*, aunque [Carroll et al., 2006] afirme que el *Activity Awareness* se incluye dentro de su definición al *Situation Awareness*. Esto puede ser si el *Activity Awareness* no se trata como un tipo de Awareness, sino como un conjunto de componentes cognitivos en los cuales está incluido el Awareness, lo cual parece ser cierto en base a la extensa descripción del *Activity Awareness* proporcionada por John Carroll.

En la Figura 18 se muestran los componentes del Awareness de Situación divididos en niveles. Primeramente, la percepción de los elementos en la situación actual. Después, la comprensión de la situación, es decir, de los elementos en conjunto. Finalmente la proyección del estado futuro de la situación, la cual

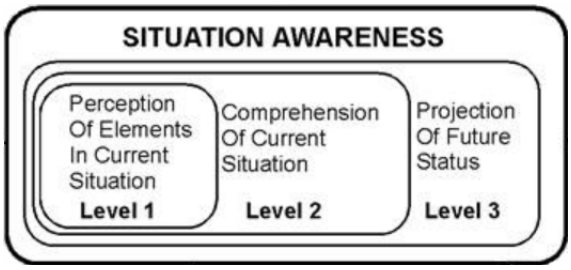


Figura 18. Componentes del Awareness de Situación: Percepción, Comprensión y Proyección.

estará representada por los elementos percibidos.

La descripción del *Situation Awareness* muestra varios aspectos clave en lo que respecta al Awareness: Percepción, Comprensión y Proyección.

La percepción implica el uso de los medios sensoriales del que requiere el Awareness, como puede ser un ser humano que usa sus sentidos o un robot que utiliza sus sensores para percibir el mundo que lo rodea. El Awareness empieza y vive por la percepción, es decir, por la percepción constante. Si una persona tapa sus ojos al caminar, no podrá ver el camino, por lo tanto pierde conciencia de su entorno exterior captado por la vista.

La comprensión implica un proceso inteligente. Se refiere a la creación de nuevo conocimiento a partir del ya existente. Por ejemplo, si uno va en el coche y detecta ruidos raros en el motor, se sabe que la situación no está bien. Lo que percibe el conductor no le da esa información directamente, sino que la utiliza y la procesa con todo su conocimiento previo, lo cual le brinda información de más alto nivel y probablemente le permita tomar mejores decisiones.

La proyección del estado de la situación es la capacidad de aproximar los valores de los elementos que definen la situación en un futuro. Esta capacidad depende de varios factores como la experiencia y las técnicas utilizadas para calcular dichos valores. Por ejemplo, para calcular las predicciones del estado del tiempo (clima) se utilizan superordenadores y gran cantidad de información real y antigua. Sin embargo, para calcular el tiempo de llegada a un lugar a velocidad

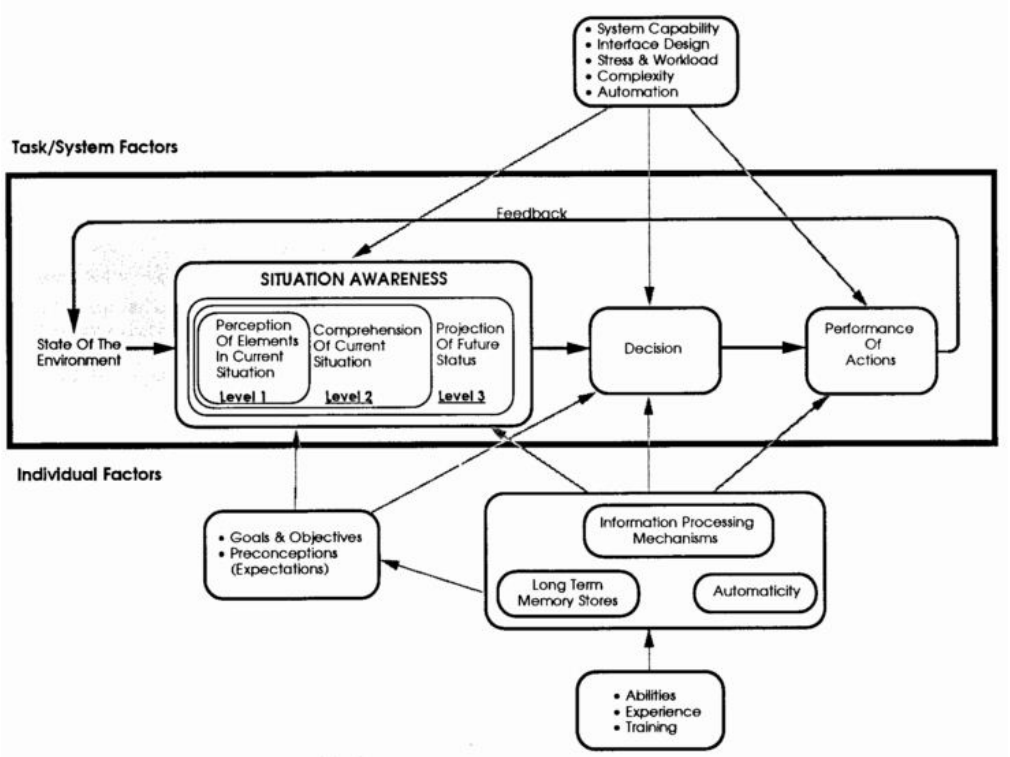


Figura 19. Diagrama del Awareness de Situación y de su papel en el proceso de toma de decisiones para la ejecución de tareas, además de los factores que intervienen en su adquisición. Adaptada de [Endsley, 1995].

constante, una fórmula física básica es suficiente.

El proceso de toma de decisiones

En la Figura 19 se muestra el diagrama del proceso de ejecución de tareas en entornos dinámicos empezando con el análisis del estado actual del entorno hasta la ejecución de la propia tarea. En este proceso interviene el *Situation Awareness* previo a la toma de decisiones. ¿A qué se debe esto? A que la decisión depende del estado actual del entorno.

Como se muestra en la Figura 19, una serie de factores afectan al proceso de toma de decisiones y a la propia adquisición del Awareness de Situación. Están los factores humanos, los factores del sistema y los factores de la tarea a desarrollar.

Factores humanos

El ciclo de ejecución/Awareness alimenta constantemente el conocimiento que se tiene del entorno y varios patrones del comportamiento pueden detectarse. Esto es lo que comúnmente se llama experiencia.

La experiencia permite relacionar los distintos elementos que componen "la situación" de interés y generar nuevo conocimiento a partir de ellas, así como aproximar los valores de dichos elementos en un futuro cercano. Sin embargo, la experiencia no sustituye al conocimiento actualizado de la situación, es decir, el *Situation Awareness*. Los dos aspectos son complementarios y aunque la experiencia puede ser muy buena, la situación puede tomar rutas inesperadas y requerir decisiones diferentes a las tomadas comúnmente.

El problema de basarse demasiado en la experiencia y dejar de lado el Awareness de la Situación se conoce como Automatización.

La automatización está fuertemente asociada a la memoria a largo plazo, en la cual se guarda el conocimiento y los patrones de comportamiento del entorno. La memoria a corto plazo, por otra parte, es también llamada memoria de trabajo y permite guardar y manipular temporalmente la información captada por los sentidos. La capacidad de la memoria de trabajo puede elevarse a través del entrenamiento, no obstante, está condicionada por el estrés y la sobrecarga cognitiva derivadas de la complejidad y dinamismo del entorno.

Factores del sistema

Un sistema informático o electrónico puede servir de intermediario entre el usuario o agente que utilizan Awareness de Situación y el propio entorno en el que se van a ejecutar las acciones. Esto puede ser un sistema computacional o el panel de control de un vehículo o nave.

El diseño de esos sistemas es un área de gran importancia para incrementar y mejorar el *Situation Awareness*, a la vez que se busca reducir la carga cognitiva

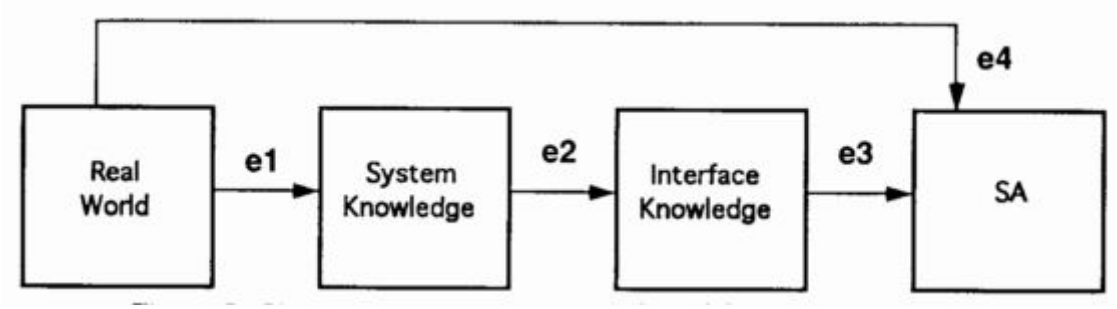


Figura 20. Flujo de información de Awarenes de Situacion: Mundo Real →Conocimiento del Sistema →Conocimiento de la Interfaz →Awareness de la Situación. Adaptada de [Endsley, 1995].

y la complejidad del entorno a través de mejores mecanismos de abstracción y transmisión de información [Goodrich et al., 2006].

En la Figura 20 se muestra un diagrama de las entradas de información que recibe un individuo para obtener un determinado Awareness de Situación cuando se utiliza un sistema intermediario entre el entorno y él mismo. La información va desde el mundo real, pasando por el sistema y por la interfaz de usuario hasta llegar al usuario. Esta serie de pasos representa el camino en el cual la información del mundo real puede perderse, malinterpretarse o unirse a otra que resulte innecesaria, por lo cual el diseño del sistema y de las interfaces de usuario forma otro eslabón en lo que es el soporte del Awareness de Situación.

Factores de la tarea

La tarea ofrece sus propias dificultades al *Situation Awareness*, como son la complejidad inherente de cada tarea, los objetivos que se busca conseguir y las precondiciones y predisposiciones aunadas a la ejecución de la tarea.

Entornos de uso y su papel en la colaboración

La investigación sobre el Awareness de Situación está orientada principalmente a los entornos altamente dinámicos y complejos, como son los entornos bélicos y de gestión de emergencias . Además de ello, el SA forma parte del

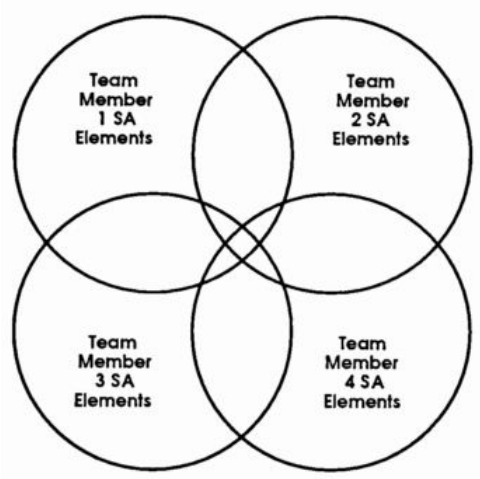


Figura 21. Awareness de Situación de un equipo de trabajo. Adaptada de [Endsley, 1995].

proceso de toma y decisiones y ejecución de tareas, por lo tanto, también es aplicable al estudio de robots, vehículos o maquinaria con capacidades autónomas [Drury et al., 2006].

En los entornos colaborativos el *Situation Awareness* es prácticamente desconocido, ya que otros investigadores han definido y documentado tipos de Awareness más específicos a entornos Groupware y CSCL, como veremos más adelante. Esto no significa que el *Situation Awareness* no pueda utilizarse en los entornos colaborativos para enriquecer la experiencia del usuario o complementar las funcionalidades requeridas por el sistema. En la Figura 21 se muestra una representación del Awareness de Situación en un equipo de trabajo.

El *Situation Awareness* que todos los miembros del equipo deben tener se representa por la conjunción del Awareness de Situación de cada uno de ellos. A su vez, el *Situation Awareness* global se representa por la suma del SA de todos. El SA grupal o común en todos los observadores permite desarrollar tareas colaborativas en las que cada ejecutor requiere tener el mismo Awareness de Situación que todos los demás sobre determinados aspectos. El SA global permite desarrollar tareas colaborativas en las que todos los individuos no tienen un SA común,

requiriendo que los propios individuos mantengan canales de comunicación para transmitirse la información necesaria durante la toma de decisiones y ejecución de tareas. Por otra parte, este tipo de interacción es base social del crecimiento humano y ciertamente permite atacar problemas más complejos y dinámicos.

Soporte y notas adicionales

Para el soporte del Awareness de Situación existe el Diseño basado en el *Situation Awareness* descrito en [Endsley et al., 2003], el cual da guías para detectar y definir los elementos del entorno que definen las situaciones de interés a cada tarea del sistema.

Debido a que las necesidades de los sistemas pueden cambiar según su propósito, el *Situation Awareness* requerido también varía. En determinados sistemas como [Matheus et al., 2003] y [Baumgartner et al., 2010] se utilizan ontologías para definir los elementos del entorno que representan la situación de interés.

El soporte del SA varía con los objetivos del sistema y depende de las necesidades específicas encontradas. A partir de estas ideas podemos entender mejor los tipos de Awareness encontrados en la literatura del Groupware y de los CSCL, como el *Workspace Awareness* que describimos a continuación.

4.2. Workspace Awareness

Para la comunidad de Interacción Persona-Ordenador (IPO) el *Workspace Awareness* es uno de los Awareness más conocidos. Reconocido como necesario para cualquier sistema Groupware, Gutwin define el *Workspace Awareness* o Awareness del Espacio de Trabajo compartido como "el conocimiento actual de las interacciones de otras personas con el espacio de trabajo compartido" [Gutwin and Greenberg, 2004].

El *Workspace Awareness* se refiere al Awareness creado durante la interacción de varios individuos en un espacio de trabajo compartido, limitándose a los eventos ocurridos en dicho espacio, ya sea asíncronos o síncronos.

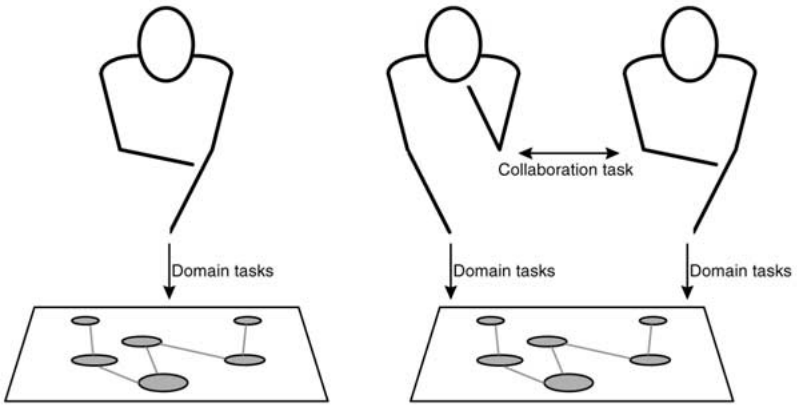


Figura 22. Tareas del dominio y de colaboración.

Descripción

El *Workspace Awareness* nace del estudio de la colaboración en los entornos informáticos. Un clásico en esta área es el trabajo de Paul Dourish sobre Awareness y Coordinación en espacios de trabajo compartidos [Dourish and Bellotti, 1992], el cual analiza el concepto de Awareness en varios sistemas colaborativos, describiendo el Awareness como "el entendimiento de las actividades de los otros, lo cual provee un contexto para nuestras propias actividades" [Dourish and Bellotti, 1992]. Esta definición es más acorde al Awareness de Actividad, descrito en la siguiente sección. Sin embargo, los puntos remarcados y analizados concuerdan mejor con el *Workspace Awareness*.

La Figura 22 muestra una metáfora de lo que representa el espacio de trabajo y las tareas del dominio a través de un sistema informático. Se puede distinguir el trabajo individual y el trabajo colaborativo a través de tareas de colaboración. Una forma sencilla de verlo es una mesa en la que puede trabajar una sola persona o varias, de tal forma que el trabajo hecho en esa mesa sirva para cumplir los objetivos del sistema.

Gutwin reconoce el *Workspace Awareness* como un caso especial [Gutwin and Greenberg, 2002] del *Situation Awareness*, el cual busca responder a las preguntas *who*, *what*, *where*, *when*, *how*, referentes a las tareas colabora-

Category	Element	Specific question
¿Quién?	Presencia	¿Se encuentra alguien en el espacio de trabajo?
	Identidad	¿Quiíen está participando? ¿Quién es ese?
	Autoría	¿Quiíen está haciendo eso?
¿Qué?	Acción	¿Qué están haciendo los demás?
	Intención	¿De qué objetivo es parte esa acción?
	Artefacto	¿Con qué artefacto están trabajando?
¿Dónde?	Localización	¿En dónde están trabajando?
	Mira	¿en donde están mirando?
	Visión	¿Qué tanto pueden ver?
	Alcance	¿Que tan lejos pueden ver y alcanzar?

Tabla 5. Elementos de información que componen el *Workspace Awareness*.

tivas que se llevan a cabo en un espacio virtual compartido. En este caso, los elementos del entorno que definen la *situación* son descritos en la Tabla 5.

Como puede observarse, los elementos que componen al *Workspace Awareness* pertenecen a varias entidades que intervienen en el proceso del trabajo colaborativo, como pueden ser los propios participantes, sus acciones, su localización, sus capacidades, etc. Esto sitúa al *Workspace Awareness* como un conjunto de Awareness de distintas situaciones, cada una de ellas importante para distintas decisiones y tareas.

El soporte del *Workspace Awareness* puede ser parcial en sus diferentes aspectos, pero normalmente se requerirá su soporte en cierta medida.

El conocimiento que representa el Awareness de Espacio de Trabajo permite una colaboración efectiva, tal y como se muestra en la Figura 23. Ayuda a simplificar la comunicación, la coordinación de las acciones y actividades, a anticipar eventos, a proveer asistencia y a administrar el grupo de trabajo. De esta forma podemos observar que el soporte del *Workspace Awareness* es necesario para el trabajo colaborativo y su soporte agrega una serie de requisitos al desarrollo del sistema, ya que los mecanismos para obtener y proveer la información que alimenta al *Workspace Awareness* deben diseñarse e implementarse con los mismos estándares de calidad que cualquier otra parte del sistema.

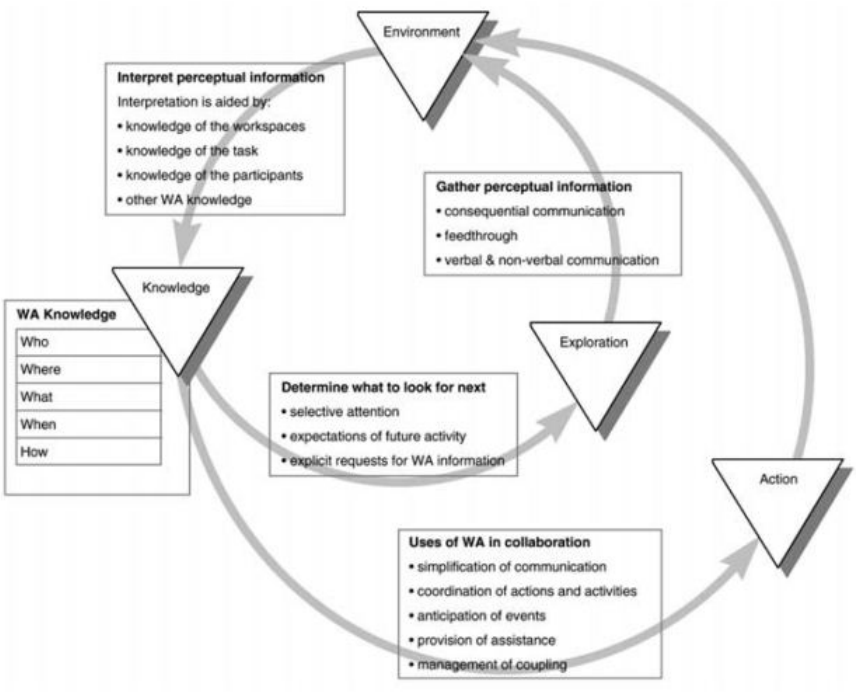


Figura 23. Ciclo de vida del *Workspace Awareness* durante el proceso de colaboración.

Entornos de uso

Como afirma [Gutwin and Greenberg, 2002], cualquier sistema que soporte tareas colaborativas a través de un espacio de trabajo compartido debe también soportar el *Workspace Awareness* adecuado a la situación y al entorno de trabajo proporcionado por el sistema.

Uno de los estudios clásicos del Awareness referente a la información de Awareness que se proporciona a los usuarios de distintos sistemas colaborativos es el previamente mencionado trabajo de Paul Dourish [Dourish and Bellotti, 1992] . Otro trabajo clásico es el del proyecto Portholes [Dourish and Bly, 1992] , un sistema colaborativo basado en la compartición de video e imágenes y pensado para construir el sentido de comunidad a distancia. Se demostró que se podía crear la sensación de Awareness a distancia, es decir, a través de un sistema informático.

Factores

El Awareness de Espacio de Trabajo, al igual que el Awareness de Situación está afectado por diversos factores externos e internos del sistema. Además, el aspecto colaborativo de estos sistemas incrementa la complejidad y la dificultad de identificar, modelar e implementar la información y los mecanismos necesarios para el correcto soporte de este tipo de Awareness.

En lo que respecta a la comunicación, en el *Workspace Awareness* se reconocen dos formas de comunicación según el tiempo: síncrona y asíncrona. Cada una de ellas requiere distintos mecanismos de Awareness debido a que la forma de interacción es distinta.

El tamaño y la jerarquía de los grupos de trabajo también afectan a los mecanismos de Awareness implementados. Por ejemplo, es distinto trabajar con grupos pequeños como en los sistemas tradicionales que trabajar con grupos de trabajo de decenas o cientos de personas, como pueden ser las redes sociales colaborativas (Facebook, Google+, etc.) o juegos en línea (World of Warcraft, Starcraft, etc').

Las interfaces de usuario juegan un papel clave en el soporte de cualquier tipo de Awareness. En el caso del *Workspace Awareness*, las interfaces deben modificarse para transmitir los aspectos inherentes a la colaboración, además de permitir una continua actualización de sus datos y ayudar a disminuir la carga cognitiva generada por los mecanismos de Awareness.

Por si esto fuera poco, cada usuario es distinto y lo que para uno funciona, para el otro no. Por tanto, se han desarrollado mecanismos de adaptación y personalización para afrontar estos problemas, aunque ciertamente el enlace entre Awareness y la toma de decisiones seguirá estando presente.

Soporte y notas extras

El *Workspace Awareness* cuenta con un excelente marco de trabajo teórico desarrollado por Gutwin [Gutwin and Greenberg, 2002], el cual funciona como guía para el soporte de este tipo de Awareness.

Con la experiencia y el análisis de distintas aplicaciones se vio la necesidad de interfaces de usuario adaptadas a los requerimientos de las tareas colaborativas. En este ámbito se desarrolló Groupkit [Roseman and Greenberg, 1996], un kit para el desarrollo de sistemas Groupware el cual incluía mecanismos para el soporte del *Workspace Awareness*. Este kit principalmente enfocado a las interfaces de usuario colaborativas fue uno de los precursores de los posteriores kits de desarrollo para sistemas colaborativos en lo que respecta a ideas base e interfaces de usuario.

[Hill and Gutwin, 2004] actualiza las ideas de Roseman y desarrolla MAUI, el cual incluye mejoras a las ideas plasmadas en Groupkit y las aplica a las nuevas plataformas tecnológicas. Sigue conservando soporte para las interfaces de usuario descritas por [Gutwin and Greenberg, 2002], como son el telepuntero, la vista de radar, las barras de movimiento multiusuario, entre otras.

En lo que respecta al desarrollo de Groupware basado en el paradigma del espacio de trabajo compartido, recientemente [Gutwin et al., 2008] presentó un nuevo enfoque para diseñar este tipo de sistemas planteando el 'Community-Based Groupware', el cual permite la colaboración informal y oportunista, la cual raramente es soportada en el paradigma tradicional. Este enfoque pretende mejorar la adopción de los sistemas Groupware, ya que se reconoce la colaboración informal como una parte esencial de los grupos de trabajo co-localizados.

El enfoque provee soporte para tres aspectos en particular: Awareness de los otros y de su trabajo individual, una manera sencilla y ligera de iniciar la interacción, y la habilidad de comenzar una colaboración estrecha y conjunta cuando se requiera.

En el soporte del *Workspace Awareness* destaca la falta de herramientas y técnicas para incluirlo en el desarrollo dirigido por modelos. Esta situación es ciertamente comprensible, ya que la naturaleza dinámica de los entornos de aplicación del *Workspace Awareness* y las distintas formas de modelar el dominio y el contexto han dificultado el desarrollo de estas técnicas.

4.3. Activity Awareness

El Awareness de Actividad está ligado al trabajo colaborativo y en cierto nivel forma parte del *Workspace Awareness*. Se refiere al Awareness generado de las actividades y tareas que se desarrollan en entornos colaborativos. El *Workspace Awareness* trata esa información de forma más centrada al conocimiento actualizado de lo que se está haciendo y quién lo está haciendo. Este enfoque es extendido en trabajos posteriores de otros autores, de tal forma que también se incluyen otros aspectos relacionados a las actividades colaborativas.

Descripción

El Awareness de Actividad como tal ha sido estudiado en mayor profundidad en [Carroll et al., 2006], el cual analiza el Activity Awareness a través de visiones diferentes mostradas en la Tabla 6, las cuales son la Base común, las Prácticas de la comunidad, el Capital social y el Desarrollo humano (Figura 24).

La base común se refiere al conocimiento compartido que se va formando a través de la colaboración, como es la historia de las actividades realizadas, las personas que las desarrollaron, los recursos implicados, las herramientas utilizados y los resultados obtenidos. Es un concepto similar al Awareness de Situación grupal, que se refiere al conocimiento que todos los miembros del equipo poseen y comparten.

Las prácticas comunitarias se refieren a las prácticas y comportamientos que son compartidos y comprendidos por todos los integrantes del grupo, como son los protocolos de comunicación, las formas de expresión, vocabulario, vestuario,

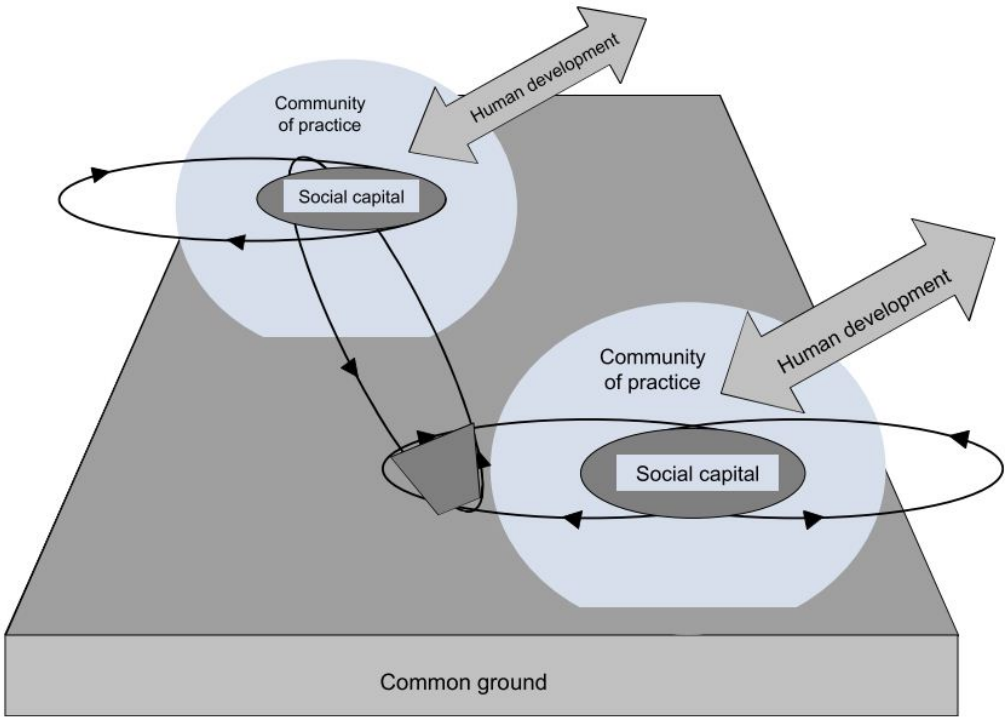


Figura 24. Cuatro facetas de la colaboración [Carroll et al., 2006].

Faceta	**Descripción**
Base común	Un protocolo de comunicación para probar y señalar conocimiento compartido y creencias.
Prácticas de comunidad	El entendimiento tácito de las conductas específicas a la comunidad compartidas a través de la práctica continua.
Capital social	La creación de habilidades y bienes persistentes a través de las redes de interacción.
Desarrollo humano	Comportamientos innovadores o decisiones derivadas de las habilidades crecientes y de la resolución de problemas complejos de cada miembro de un equipo y del equipo como conjunto.

Tabla 6. Cuatro facetas del Awareness de Actividad en la colaboración soportada por ordenador [Carroll et al., 2006].

entre otros.

El capital social se refiere a las relaciones sociales generados por una buena interacción, es decir, relaciones que les permiten a los miembros colaborar mejor cuando están juntos, en vez de que generen estrés o problemas de comunicación y coordinación durante la interacción. Estas relaciones humanas se generan tras un tiempo de colaboración y adaptación mutua, en las cuales los miembros del equipo aplican sus capacidades sociales y profesionales para trabajar en conjunto con otros, formando de esta formas esas preciadas relaciones que permiten un trabajo colaborativo efectivo y beneficioso para todos.

El desarrollo humano se refiere al aprendizaje de nuevo conocimiento y habilidades, tanto humanas como profesionales, el cual con el tiempo también se vuelve un desarrollo humano compartido.

El Activity Awareness puede verse como un modelado del Awareness requerido para realizar actividades colaborativas, a la vez que toma en cuenta el proceso de evolución de todos los integrantes del sistema. Parte de puntos de vista sociológicos y crea un modelo muy distinto a los anteriormente vistos, como los de Endsley y Gutwin. Sin embargo, este Tipo de Awareness complementa los otros marcos de trabajo con su visión social y evolutiva de lo que son las tareas colaborativas.

Entornos de uso

El Activity Awareness se aplica principalmente al trabajo colaborativo, de tal forma que permita a los miembros del equipo coordinar sus actividades. También puede utilizarse en entornos en donde la realización de las tareas dependa en gran medida del estado del entorno.

Las cuatro visiones del Awareness de Actividad pueden apoyarse por otros tipos de Awareness específicas para ese objetivo. Es decir, para apoyar a los miembros del equipo en el desarrollo de estas habilidades de forma individual y grupal.

Soporte y notas adicionales

El soporte del Awareness de Actividad puede analizarse desde dos puntos de vista. El primero es brindar la información para soportar la colaboración durante el trabajo grupal. El segundo es brindar la información sobre las actividades actuales de cada miembro del equipo, así como información de cada actividad en su contexto de uso.

Muchas de las interfaces propuestas para el soporte del *Workspace Awareness* se utilizan para dar soporte en la UI al Activity Awareness. Sin embargo, dependiendo de las necesidades de cada sistema, cada interfaz y mecanismo de distribución de datos pueden especializarse para cumplir el propósito deseado.

4.4. Context Awareness

Este tipo de Awareness referente al contexto va comúnmente ligado a los sistemas móviles y ubicuos.

El contexto es lo que rodea a un objeto, entidad, idea o concepto, de tal forma que brinde un marco de referencia para darle significado más específico y evitar ambigüedades.

Es tal vez el tipo de Awareness más popular actualmente. Esto se debe a su relación con las tecnologías móviles [Kirsch-Pinheiro et al., 2005], la inteligencia ambiental [Cook et al., 2009] y la computación ubicua [Baldauf et al., 2007].

El Awareness del Contexto no es exclusivo de estas áreas, ya que el concepto de Contexto se aplica a "lo que rodea a un objeto o idea" y cambia dependiendo de los objetivos de esa información. Por ejemplo, un contexto puede ser útil durante la búsqueda de hoteles, pero otro puede ser más útil durante la búsqueda de restaurantes.

Descripción

Definir qué información identifica el contexto de utilidad depende de los objetivos de los usuarios. Esto puede confundirse con el Awareness de Situación, ya que en los dos se requiere identificar los elementos del entorno que definen la *situación* de interés.

El contexto es la parte de la situación que rodea el objetivo o idea principal que se está desarrollando, ya sea física o conceptualmente. Su labor es darle un significado preciso a cierta información, evitando interpretar de formas distintas el mismo concepto.

Por ejemplo, hablar de la caza en el contexto de la caza de animales, o hablar de la caza en el contexto de la búsqueda de errores en un software. Conocer el contexto nos permite situarnos en el espacio cognitivo correcto.

El Awareness de contexto puede reconocerse como un extra para la el entendimiento del entorno de interés, y como tal, representa un tipo de Awareness más que debe ser soportado por el sistema, lo cual genera la necesidad de definir la información del entorno que va a representar el contexto de interés para una determinada situación.

Para cuestiones prácticas, el Awareness de contexto representa un tipo de Awareness ligado a otros tipos de Awareness, de tal forma que a través del mismo el observador pueda entender mejor la información percibida en ese contexto y situación.

Entornos de uso

El Context Awareness se maneja principalmente en sistemas para dispositivos móviles [Bales et al., 2011], computación ubicua [Baldauf et al., 2007], inteligencia ambiental [Cook et al., 2009], entre otros [Feng et al., 2009].

En las aplicaciones móviles el Context Awareness se utiliza para proporcionar información relacionada a la localización física, como pueden ser sitios de interés,

actividades, personas o cualquier otro conocimiento previamente definido como *de interés* para el portador del dispositivo.

En la computación ubicua el Context Awareness juega un papel similar al de las aplicaciones móviles. Sin embargo, el contexto de interés puede ser más amplio y servir no sólo al portador del dispositivo, sino al propio dispositivo.

En la domótica el contexto de utilidad puede referirse a las condiciones físicas internas y externas, a las actividades programadas, a las personas que están o no están en las instalaciones, etc.

El Contexto es importante para otras aplicaciones como la búsqueda de información. Por ejemplo, uno de los problemas típicos durante la búsqueda de información es que normalmente depende del contexto del usuario que la va a utilizar, ya que dependiendo de este contexto, la información recabada puede ser o no útil [Tsai et al., 2010].

Una de las aplicaciones indirectas del Context Awareness es la adaptación de contenidos según el contexto físico, tecnológico o conceptual. La adaptación de contenidos es necesaria debido a la diversidad tecnológica de los dispositivos móviles y a la diversidad de los diferentes usuarios de estos dispositivos. La adaptación busca mejorar la experiencia de usuario al aprovechar al máximo las capacidades y condiciones del dispositivo. También se puede aprovechar las condiciones y habilidades del portador de tal manera que la información recibida sea más útil y asimilable [Krause et al., 2006].

Soporte y notas adicionales

Las herramientas para el soporte del Context Awareness van de la mano de la popularidad de los dispositivos móviles. Los teléfonos y otros dispositivos móviles están eclipsando al ordenador en lo que respecta a uso generalizado. Por tanto, no es de extrañar que el soporte del Context Awareness sea un tema de interés para la comunidad. Además, las aplicaciones del Context Awareness no están limitadas a las tareas individuales, sino que también puede participar en

entornos colaborativos a través de redes sociales y juegos en los que el contexto individual interviene en las actividades colaborativas.

El trabajo de [Baldauf et al., 2007] analiza distintos enfoques para soportar el Context Awareness en distintos sistemas y se enfoca en los marcos de trabajo y middlewares para este propósito. Un aspecto importante de este análisis y la observación de la gran variedad de representaciones del contexto, produciendo un problema de incompatibilidad entre formatos de representación. Por otra parte el trabajo de [Strang and Popien, 2004] muestra que uno de los enfoques más prometedores para representar el contexto en un sistema es el basado en ontologías, ya que es el más completo en lo que respecta a flexibilidad y capacidad.

Un aspecto muy interesante del Context Awareness es la posibilidad de captar y transmitir información sobre el estado de ánimo o estado físico del portador del dispositivo. Esto puede enriquecer en gran medida los sistemas Groupware al agregar una dimensión personal e individual al conjunto de representaciones o avatares que puede tener un miembro del equipo [Neyem et al., 2007].

4.5. Peripheral Awareness

Como se ha visto anteriormente, varios factores afectan la formación del Awareness. La capacidad de atención restringe la cantidad de información que puede asimilarse. Si se proporciona información por encima de la capacidad de atención se produce una sobrecarga cognitiva, la cual impide la formación del Awareness.

Por otra parte, cuando una interfaz de usuario de Awareness requiere más atención que la tarea principal, entonces se produce la distracción.

Ninguno de estos efectos es deseable en un sistema que proporcione Awareness. Sin embargo, esta situación es inherente al soporte del Awareness. La idea del Awareness periférico o Peripheral Awareness es complementar los mecanismos de Awareness principales y la cantidad de información transmitida al usuario, así como disminuir la carga cognitiva necesaria para ello, sin reemplazar o congestionar los canales principales de comunicación. También es importante reducir

los errores de interpretación u omisión, así como incrementar la satisfacción del usuario con los mecanismos de Awareness [Matthews et al., 2007].

Descripción

El Awareness periférico no se refiere a un tipo específico de información, sino al método utilizado para transmitirla o a los medios receptores objetivo [Pedersen and Sokoler, 1997]. En el caso de los sistemas informáticos comunes, los mecanismos disponibles para transmitir información son la visión (pantalla del dispositivo) y el audio (altavoces). Sin embargo, se pueden utilizar otras formas de transmisión que exploten otros sentidos del ser humano, como es el tacto, el olfato, la sensación de frío o calor, la presión, el olfato, etc.

Usando los medios convencionales de transmisión de información se puede ofrecer el Peripheral Awareness. No obstante, las interfaces de usuario para este propósito buscarán utilizar mecanismos más primitivos o 'pre atención'. Por ejemplo, [Treisman and Paterson, 1984] reconoce el uso de formas, colores, movimientos, ritmos, melodías, luces, etc. Se dice que estos sentidos son periféricos porque no ocupan toda la atención del observador para ser captados. Es decir, permite una transmisión de los datos de Awareness menos exigente en el sentido cognitivo, siendo de esta forma un apoyo a los mecanismos de transmisión principales.

En cuestión de información, el Awareness periférico puede utilizarse para transmitir otros tipos de Awareness menos importantes o para transmitir el Awareness del Contexto [Holmquist et al., 1999].

Entornos de uso

La forma más común de proporcionar el Peripheral Awareness es a través de los canales de comunicación convencionales, es decir, interfaces gráficas. Sin embargo, para no saturar el canal de comunicación se utilizan técnicas básicas como el uso de colores, movimientos, formas, etc.

Los entornos de uso del Awareness periférico son muy variados. Sin embargo, los objetivos permanecen: reducir la carga cognitiva y transmitir información complementaria.

Un sistema clásico cuyo objetivo fue probar el Peripheral Awareness es AROMA [Pedersen and Sokoler, 1997]. Explora el uso de representaciones abstractas como indicadores de presencia, siendo el objetivo principal el explorar el tipo de datos que deben capturarse y desplegarse para proporcionar la sensación de presencia remota. Las representaciones abstractas en este entorno proporcionan información sin requerir toda la atención del usuario y abarcar distintos sentidos como la visión, el oído, el calor y la humedad. Todo ello con el objetivo de hacer sentir a los usuarios como si estuvieran co-localizados.

La información del contexto personal del usuario puede transmitirse por interfaces periféricas. Sideshow [Cadiz et al., 2001] de Microsoft® utiliza una barra lateral para distribuir información personalizada por el propio usuario, como pueden ser sus compañeros y amigos en línea, redes sociales, el clima o el tráfico, noticias, entre otros. El uso de avisos o mensajes es reducido para evitar las interrupciones. No obstante, permite notificaciones de diversos tipos como son los colores, el parpadeo o pequeñas ventanas emergentes.

Utilizando el enfoque multi sensorial, se utilizan distintos mecanismos y canales para transmitir la misma información. Esto disminuye el grado de error de interpretación y captura que se puede tener en una aplicación determinada. Por ejemplo, en algunas aplicaciones es vital conocer el estado actual del suministro del aire, por ejemplo, en minas. Para transmitir este Awareness pueden utilizarse interfaces gráficas de usuario, interfaces de voz y tal vez dispositivos portables vibratorios, termales u otros. Los juegos en realidad virtual también pueden beneficiarse del Awareness Periférico multi sensorial al transmitirle al usuario información vital a través de diversas interfaces.

Los teléfonos móviles aprovechan un mecanismo de comunicación muy particular y prácticamente omnipresente en todos ellos: la vibración. Varios disposi-

tivos incluyen esta interfaz de usuario, la cual estimula el sentido del tacto para transmitir información. Esta interfaz aprovecha la duración de la vibración, la intensidad, los espacios entre vibraciones y el ritmo formado por todos ellos, hasta el punto de poder reproducir música a través de estas vibraciones.

Los acelerómetros encontrados en algunos dispositivos móviles como teléfonos o tablets permiten capturar el movimiento del propio dispositivo, aprovechado comúnmente en juegos para el propio dispositivo, pero también utilizable como apoyo a tareas individuales o colaborativas [Luqman and Griss, 2010].

Soporte y notas adicionales

El soporte del Awareness Periférico se divide en dos partes. Primero, identificar el tipo de información a transmitirse. La segunda y más notoria es elegir las interfaces de usuario para transmitir dicha información. Estas interfaces deben producir muy poca sobrecarga cognitiva, no deben ocupar toda la atención del usuario y deben tener una alta efectividad en lo que respecta a transmitir información sin errores.

El desarrollo de las interfaces de usuario para proporcionar Awareness periférico es un área compleja en la que se encuentran las capacidades tecnológicas, las capacidades sensoriales humanas (o del ente que recibe la información) y el aspecto psicológico o tecnológico que afecta la asimilación de la información por esos canales de pre-atención [Treisman and Paterson, 1984].

En lo que respecta a la evaluación de este tipo de interfaces de usuario, [Matthews et al., 2007] presenta un marco de trabajo para definir, diseñar y evaluar interfaces de usuario periféricas, en el cual define el Awareness como uno de los aspectos a valorar en dichas interfaces.

El mundo de las interfaces de usuario sigue evolucionando y cada vez más nos encontramos con interfaces de usuario que salen de lo gráfico a lo tangible [Ishii, 2008]. Algunas de estas interfaces de usuario tendrán capacidades para transmitir Awareness y otras no. Prácticamente la única capacidad necesaria

para transmitir Awareness es la capacidad de actualizar alguna de sus características, como el tamaño, color, brillo, luz o luces, temperatura, movimiento, entre otras. Teniendo alguna de esas capacidades le permite transmitir información de Awareness (actualizada y continua).

4.6. Knowledge Awareness y Shared-Knowledge Awareness

Estos tipos de Awareness se utilizan en entornos de aprendizaje colaborativo, aunque su propósito no es el mismo. Es importante diferenciar el conocimiento individual y el conocimiento grupal, el cual es la suma de todo el conocimiento individual. Por otra parte, el conocimiento compartido es el conocimiento igual y compartido que cada individuo del grupo tiene, es decir, lo que todos conocen. El Awareness del Conocimiento [Ogata et al., 1996] se orienta al conocimiento individual y grupal y el Awareness del Conocimiento Compartido [Collazos et al., 2003] o Shared-Knowledge Awareness (SKA) se orienta al conocimiento compartido y a su construcción.

Descripción

El Awareness del Conocimiento según [Ogata et al., 1996] permite conocer los siguientes aspectos sobre el conocimiento generado en el sistema: ¿Quién está buscando conocimiento igual o diferente al que yo estoy buscando?, ¿Quién está discutiendo conocimiento igual o diferente al que yo estoy discutiendo? Y ¿Quién está cambiando (agregando/borrando/modificando) conocimiento? Tabla 7.

Las cuestiones de la Tabla 7 se refieren al conocimiento que el usuario está utilizando y al que no está utilizando. El contexto de uso puede complementar este tipo de Awareness, ya que normalmente en aprendizaje se genera durante actividades individuales o colaborativas, además de permitir la adaptación de contenidos al nivel de los estudiantes [Yang, 2006].

El Awareness del Conocimiento Compartido se enfoca a la construcción del conocimiento individual y a la construcción del conocimiento compartido.

	Mismo Conocimiento	Conocimiento Diferente
Mismo Tiempo	¿Quénes están discutiendo el conocimiento? ¿Quíenes están buscando conocimiento? ¿Quíenes están cambiando el conocimiento?	¿Qué conocimiento están discutiendo? ¿Qué conocimiento están buscando? ¿Qué conocimiento están cambiando?
Diferente Tiempo	¿Quiénes discutieron sobre el conocimiento? ¿Quiénes buscaron conocimento? ¿Quiénes cambiaron el conocimento?	¿Qué conocimiento discutieron? ¿Qué conocimiento buscaron? ¿Qué conocimiento cambiaron?

Tabla 7. Tipos de mensajes del *Knowledge Awareness* [Ogata et al., 1996].

[Collazos et al., 2003] define el SKA a partir del aprendizaje generado por la realización de tareas colaborativas, ya sea por aprendizaje o por trabajo colaborativo y por la interacción entre los miembros del equipo. La Tabla 8 muestra una descripción de lo que representa el SKA y el tipo de conocimiento actualizado que proporciona, de tal forma que los usuarios puedan interactuar mejor y realizar más efectivamente las tareas conjuntas.

En el caso del SKA, la relación entre el conocimiento generado y el entorno de trabajo o aprendizaje está mejor definida. En la Figura 25 se muestra la relación entre el espacio de trabajo compartido, el aspecto social, la tarea y el concepto y el Conocimiento compartido.

Entornos de uso

Está claro que estos tipos del Awareness benefician a los entornos de aprendizaje y en algunos casos a los sistemas Groupware en general. Por ejemplo, los sistemas académicos son candidatos adecuados para estos tipos de Awareness. También los sistemas Groupware orientados a la capacitación y entrenamiento de personal pueden beneficiarse de este conocimiento.

Un entorno en el que se mezcla el trabajo colaborativo y la compartición

Awareness	**Preguntas**
Construcción del *Conocimiento* (individual)	¿Lo que estoy haciendo está ayudando a resolver la tarea? ¿Lo que hice está ayudando a resolver la tarea? ¿Necesesito más tiempo/recursos? ¿Qué necesito saber sobre este tema? ¿Qué más necesito encontrar sobre este tema? ¿De qué tanto tiempo dispongo? ¿Cúal es nuestro puntaje? ¿Qué y cómo aprendí de los otros miembros del grupo? ¿Terminé mi trabajo? ¿Qué es lo que estoy aprendiendo de este trabajo en equipo?
Construcción del *Conocimiento Compartido* (grupal)	¿Qué es lo que están haciendo los otros miembros del equipo para completar la tarea? ¿Las actividades de los otros miembros del equipo están ayudando a resolver la tarea? ¿Qué es lo que los otros miembros del equipo conocen sobre el tema? ¿Qué es lo que los otros miembros del equipo necesitan conocer sobre la tarea? ¿Cómo puedo ayudar a los otros estudiantes a completar la tarea? ¿Qué es lo que los otros miembros del equipo han aprendido de mí? ¿Dónde están los otros miembros del equipo?

Tabla 8. Awareness del Conocimiento Compartido o *Shared-Knowledge Awareness* [Collazos et al., 2003].

Figura 25. Relación del SKA con el entorno de aprendizaje o de trabajo colaborativo [Collazos et al., 2003].

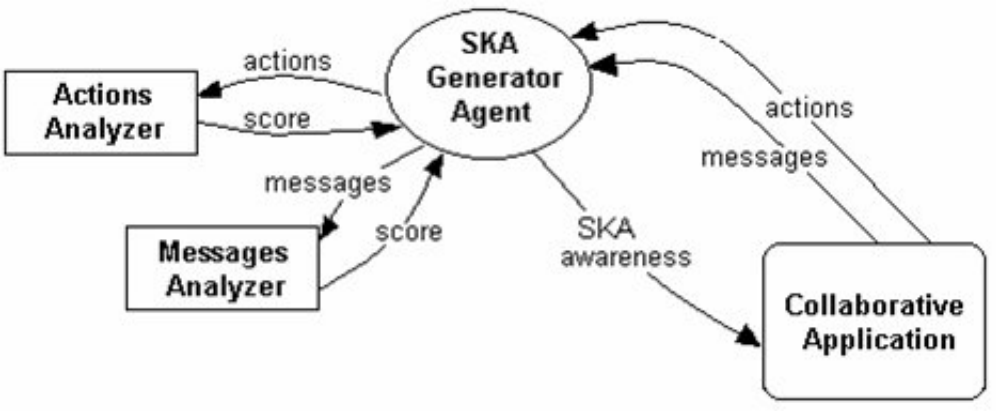


Figura 26. Mecanismo para el soporte del Shared-Knowledge Awareness [Collazos et al., 2003].

de conocimiento es la traducción colaborativa asistida por ordenador. El trabajo presentado por [Yong, 2009] muestra la importancia de varios tipos de Awareness durante esta actividad, siendo directamente utilizado el *Workspace Awareness* e indirectamente el Knowledge Awareness.

Soporte y notas adicionales

Tal vez la mayor dificultad para el soporte del Knowledge y Shared-Knowledge Awareness es identificar cuándo se crea, modifica o elimina el conocimiento de interés para el sistema. Sharlok [Ogata et al., 1996] utiliza TRIAS [Yamamoto et al., 1989] para representar el conocimiento en forma de triadas, algo parecido al modelo de datos RDF (Resource Description Framework)[4].

La propuesta de [Collazos et al., 2003] para el soporte del SKA al igual que el trabajo de [Farooq et al., 2007] sobre el Awareness para el soporte de la creatividad en sistemas distribuidos utiliza un analizador de mensajes para detectar el conocimiento transmitido entre los miembros del equipo. La Figura 26 muestra el esquema completo del mecanismo para el SKA.

En las propuestas que utilizan el análisis de mensajes para detectar el intercambio de conocimiento sigue siendo necesaria una forma de representar ese

[4] `http://www.w3.org/TR/rdf-primer/`

conocimiento, a fin de poder transmitir el Awareness adecuado dependiendo de cómo se utiliza dicho conocimiento.

Las ontologías pueden utilizarse para representar y almacenar conocimiento o "la situación", en el caso del *Situation Awareness* [Baumgartner et al., 2010].

4.7. Presence, Availability, Schedule, Rhythm Awareness

Las actividades colaborativas pueden realizarse al mismo tiempo (actividades síncronas) o en tiempo distinto (actividades asíncronas). Dependiendo de esto el estilo de colaboración será distinto, al igual que deben ser distintos los mecanismos software que soportan dicha colaboración. Saber si un miembro del equipo de trabajo está disponible determinará la forma de trabajar en la actividad colaborativa.

En la literatura científica del Awareness es común encontrar referencias a la Presencia y Disponibilidad. Presencia significa que el usuario tiene una sesión abierta en el sistema. Sin embargo, esto no implica Disponibilidad, ya que puede estar realizando otras actividades que requieren su completa atención.

El Schedule Awareness [Kim and Kim, 2007] se refiere al Awareness de los tiempos de Presencia en el sistema, como pueden ser los horarios programados o dinámicos del usuario. La Disponibilidad dependerá de otros factores. Los calendarios para reuniones como Google Calendar permiten coordinar reuniones basándose en la disponibilidad de horario de la mayor cantidad de miembros de un equipo. Si esta información se agrega al espacio de trabajo compartido, entonces todos los miembros del equipo pueden estar al tanto de dichos horarios.

El Rhythm Awareness o Awareness del Ritmo es una propuesta de [Begole and Tang, 2007] que enfatiza la importancia de la disponibilidad y el ritmo de disponibilidad en el diseño de aplicaciones colaborativas. En este caso, el ritmo de disponibilidad es una estimación de los momentos en los que cada miembro del equipo estará disponible. Esto es útil por ejemplo, para concertar citas o para programar reuniones. La estimación de los tiempos de disponibilidad

se hace a partir de algoritmos diversos y de los datos históricos de la aplicación referentes a la disponibilidad que cada usuario ha tenido.

Los tiempos de Presencia son un componente habitual en los sistemas más populares de mensajería instantánea, como son Microsoft Live Messenger, Google Talk, Skype, etc. Su importancia es clara ya que si el usuario no está Presente significa que no puede haber una interacción síncrona con él y si se desea una interacción tendrá que hacerse por medios asíncronos, como es el correo electrónico, por ejemplo.

El soporte de este tipo de Awareness está ligado al manejo de sesiones y al manejo de historiales de conexión, siendo casi omnipresente en la gran mayoría de las aplicaciones actuales de mensajería, trabajo colaborativo, redes sociales, foros y sistemas de aprendizaje colaborativo.

4.8. Change Awareness

A partir de esta sección se otros tipos de Awareness menos utilizados pero que aparecen en la literatura científica y que son útiles para mostrar los distintos usos del Awareness en los sistemas Groupware.

El Change Awareness o Awareness del Cambio propuesto por [Tam and Greenberg, 2006] se refiere al Awareness de los cambios asíncronos hechos a un documento colaborativo. Tam [Tam and Greenberg, 2006]incluye una serie de preguntas de alto nivel para definir el tipo de información que el Change Awareness debe ayudar a responder:

1. ¿Dónde se han hecho cambios?
2. ¿Quén ha hecho cambios?
3. ¿Qué cambios se han hecho?
4. ¿De qué forma han cambiado las cosas?
5. ¿Dónde sucedieron los cambios?

6. ¿Porqé se hicieron los cambios?

Al requerir el Change Awareness en el desarrollo de tareas colaborativas (edición colaborativa de documentos) también se requieren otros tipos de Awareness como el *Workspace Awareness*. Claramente el nombre propuesto para este tipo de Awareness es muy corto y genérico. En la contra parte, nombres más específicos pueden llegar a ser muy largos. Por ejemplo, para este caso, el Change Awareness propuesto puede presentarse como "Asyncronous Document Change Awareness". El nombre no es tan largo y resulta mucho más descriptivo.

4.9. Anticipate Awareness

El tipo de Awareness llamado Anticipate Awareness estudiado en [Prinz et al., 2010] se refiere a las expectativas que se pueden tener sobre las acciones de los otros miembros del equipo. Es decir, tener un Awareness del futuro, difiriendo de los tipos de Awareness comunes que se enfocan en el presente y pasado.

4.10. Emotional Awareness

En el trabajo de [Neyem et al., 2007] se proponen un conjunto de interfaces de usuario y mecanismos para proporcionar Emotional Awareness o Awareness de la emoción. Este tipo de Awareness se centra en la transmisión de la información referente a las emociones del usuario y de los demás miembros del grupo de trabajo. En este caso, el nombre de este tipo de Awareness permite conocer su objetivo y de alguna forma, su utilización como medio para mejorar la interacción entre usuarios colaboradores.

4.11. Difficulty Awareness

Otras propuestas no especifican un nombre característico al tipo de Awareness que maneja su contribución. Por ejemplo, en el trabajo de [Carter and Dewan, 2010] enfocado a la mejora de la enseñanza o entrenamiento

de desarrolladores de software, se utiliza el conocimiento de los miembros que tienen dificultad con algún algoritmo. No se propone un nombre en particular para este tipo de Awareness, sino que se remarca su utilidad en estos entornos.

Darle nombre a un tipo de Awareness no es tarea fácil. Actualmente se siguen prefiriendo nombres cortos y genéricos para nombrarlos. Sin embargo, en entornos de desarrollo en donde exista la posibilidad de implementar varios tipos de Awareness de la misma familia, los nombres para dichos tipos de Awareness similares tendrán que cambiarse para ser más específicos y no crear confusiones a los desarrolladores y usuarios del software.

5. Soporte del Awareness en el desarrollo de software

La inclusión del concepto de Awareness en el software interactivo ha sido posible gracias a varias herramientas y técnicas conceptuales y técnicas las cuales, aunque pueden diferir conceptual y objetivamente, permiten incluir en el desarrollo del sistema la capacidad de proporcionar alguna forma de Awareness.

En esta Sección describiremos el soporte del Awareness en distintas partes del proceso de desarrollo del software. Algunas de las herramientas o técnicas presentadas no tienen una relación directa o pueden no utilizarse conjuntamente. Las metodologías especializadas en los sistemas Groupware ofrecen un entorno más unificado en este aspecto. Sin embargo, varias de los trabajos revisados en esta sección nos han permitido conocer como otros investigadores y desarrolladores entienden el Awareness y que tipo de técnicas utilizan para soportarlo.

5.1. Soporte teórico y conceptual

Cualquier mecanismo, artefacto, marco de trabajo o interfaz de usuario para darle soporte al Awareness tiene una base teórica sobre la cual apoyarse. Si un sistema proporciona algún tipo de Awareness es porque de antemano los desarrolladores reconocen su importancia.

Las diversas teorías sobre el Awareness permiten comprenderlo desde la perspectiva de la propia teoría y de alguna forma sirven como guía para su soporte en el software.

A continuación describimos algunas de las teorías base sobre el Awareness desde el punto de vista del desarrollo del software.

5.1.1. *Teoría del Awareness de Situación*

La Teoría del Awareness de Situación o Situation Awareness (SA) [Endsley, 1995] describe el Awareness de Situación, su papel en el proceso de toma de decisiones y los factores que afectan su formación. Ver la Sección 4.1.

A través de esta teoría se entiende que para dar soporte a la toma de una decisión que requiera Awareness, se debe proporcionar la información de la situación de interés al nivel de Awareness requerido por el usuario o usuarios, tomando en cuenta los factores que afectan la asimilación de dicha información y los factores del sistema que afectan la extracción y transmisión de la información de Awareness.

A nivel general, el Awareness se soporta en un sistema software al proporcionar información actualizada sobre una determinada situación o hecho, de tal forma que se genere ese estado mental de conocimento actualizado necesario para la toma de decisiones.

El soporte del Situation Awareness comienza con detectar la situación de interés de la cual se requiere Awareness. Esta es una tarea necesaria para prácticamente cualquier sistema, ya que al cambiar el dominio y las decisiones a tomar, también cambia la información necesaria para proporcionar dicho Awareness.

Endsley propone el Diseño Orientado al Awareness de Situacion [Endsley et al., 2003] para descubrir y definir la información de Awarenes utilizada para las decisiones que permiten lograr un objetivo en el sistema. De esta forma se busca resolver la primera barrera para el soporte del SA, la cual es

definir qué situación es la que se requiere conocer de forma actualizada.

Los factores que afectan la generación del SA en los humanos son bien tratados en este estudio, destacando la sobrecarga cognitiva, la cual se resume en la sentencia "*más datos no es igual a más información*". Este factor implica que la información que se transmite puede ser poca o demasiada dependiendo del usuario que la reciba. Por lo tanto, es preferible que este factor se tome en cuenta cuando se diseñan sistemas con soporte del SA.

Un factor muy limitante en la generación de cualquier Awareness es que sólo pueden obtenerse los datos que el sistema proporcione. Es decir, los diseñadores del software deben especificar cuales son los datos de la situación que son importantes, ya que son los únicos que van a poder ser percibidos por los usuarios. ¿Qué tal si no son suficientes o si son demasiados? Esta incógnita puede afectar toda la estructura del sistema, desde la definición de la situación, la captura y manipulación de los datos hasta las interfaces de usuario que los proveen, pasando por los marcos de trabajo, kits de desarrollo, control de acceso, etc.

Casi todas las partes del sistema pueden verse afectadas si el Awareness de Situación no es definido correctamente o si cambia durante el proceso de desarrollo. Esto es algo para tenerse en cuenta.

Para este trabajo consideramos la Teoría del Awareness de Situación como la más completa y descriptiva, aunque no está orientada al desarrollo de software ni a los sistemas colaborativos. Sin embargo, puede especializarse con la información proporcionada por los otros trabajos mostrados a continuación.

5.1.2. *Marco descriptivo del Awareness de Espacio de Trabajo Compartido*

El marco descriptivo del Awareness de Espacio de Trabajo Compartido o Workspace Awarenes (WA) desarrollado en [Gutwin and Greenberg, 2002] pretende ser una guía para ayudar a los desarrolladores de sistemas colaborativos a soportar el WA (Ver Sección 4.2).

Este marco descriptivo describe el paradigma del espacio de trabajo compartido, el cual es un punto de interacción común en tiempo real de diversos usuarios, los cuales buscan cumplir un objetivo mayor en común. Para colaborar en este entorno se requiere conocer al minuto diversa información referente a los usuarios, sus actividades y los objetivos de cada uno de ellos. Esta información es descrita de forma general en este marco de trabajo.

Según Gutwin, el WA es una forma especializada del SA, siendo utilizado en entornos de trabajo colaborativo. De esta forma, todos los elementos que Gutwin define como parte del WA definen la situación de interés para los usuarios de sistema.

El WA está formado por el Awareness de los elementos de varias entidades; por ejemplo, los integrantes del grupo, las actividades de cada uno de ellos, su localización, etc. Estos elementos no pertenecen a una sola entidad, ni tampoco pueden definirse de forma única para cualquier dominio, ya que dependen de la estructura de las entidades del dominio y del sistema. Por lo tanto, el WA es un conjunto de Awareness más localizados los cuales brindan en conjunto el WA.

El WA es el tipo de Awareness más conocido en la comunidad del Groupware y es que tiene bastante investigación de fondo. Además, muchos procesos colaborativos se llevan a cabo en tiempo real y usan un espacio de trabajo compartido para ello, con lo cual, el WA es requerido.

Esto no implica que el WA sea el único Awareness requerido cuando se soporta la colaboración en tiempo real usando espacios compartidos. Pueden necesitarse otros tipos de Awareness dependiendo de los objetivos del sistema y de las decisiones necesarias para lograrlos. Por ejemplo, un entorno de aprendizaje colaborativo normalmente requerirá Awareness específicos para este proceso, además del WA en caso de hacerlo en espacios compartidos.

Una parte importante del WA es la información sobre las actividades individuales y conjuntas. Esta información es de gran importancia para el trabajo

colaborativa y su estudio es ampliamente abordado en los trabajos de John Carroll [Carroll et al., 2003, Carroll et al., 2006], presentados a continuación en un marco descriptivo sobre el Awareness de Actividad.

5.1.3. *Teoría del Awareness de Actividad y el trabajo de equipo*

John Carroll presenta en [Carroll et al., 2006] un trabajo muy bien fundamentado sobre el Awareness y su repercusión en la efectividad y el desarrollo del trabajo en equipo. Carroll analiza el trabajo en equipo como una actividad y se refiere al Awareness de actividad como la necesidad de compartir (en el contexto del trabajo en equipo).

Según este estudio, para soportar el Awareness de actividad es necesario soportar las 4 facetas del trabajo en equipo, las cuales son:

- **Base común**: Un protocolo de comunicación para probar y señalar conocimiento compartido y creencias.
- **Prácticas de comunidad**: El entendimiento táctico de las conductas específicas a la comunidad compartidas a traés de la práctica continua.
- **Capital social**: La creación de habilidades y bienes persistentes a través de las redes interacción.
- **Desarrollo humano**: Comportamientos innovadores o decisiones derivadas de las habilidades crecientes y de la resolución de problemas complejos de cada miembro de un equipo y del equipo como conjunto.

El soporte de cada una de estas facetas del trabajo en equipo se traduce en proveer información relevante sobre cada una de las facetas a los usuarios dependiendo de sus actividades y necesidades en el momento.

Siguiendo el enfoque de que proveer Awareness implica proveer información para generarlo, el soporte de cada una de las facetas del Awareness de actividad puede plantearse como proveer la información que permita su generación:

- **Base común**: La información referente al conocimiento que tienen los otros colaboradores sobre el entorno y las entidades relacionadas con el mismo; es decir, que tanto saben y cual parte de ese conocimiento es importante para la tarea actual.

- **Prácticas de comunidad**: La información referente a las prácticas comunes a todos los miembros del equipo, ya sea como costumbre o protocolo. Esta información le permite a los nuevos integrantes del equipo integrarse mejor a las prácticas normales del grupo e incrementa su capacidad de colaboración con los mismos.

- **Capital social**: La información referente a las actividades que puden combinarse o realizarse para lograr el objetivo más rápido, con menos recursos o con menos riesgo. De esta forma se pueden crean enlaces sociales más efectivos y unir a los miembros del equipo.

- **Desarrollo humano**: La información sobre las capacidades y habilidades de interés de los miembros del equipo. Por ejemplo, saber quien es más experto en la realización de una tarea de riesgo, o quien tien una lesión o impedimento físico, técnico o diverso.

Esta claro que la información de Awareness para soportar cada faceta del trabajo en equipo depende del dominio de aplicación, del trabajo que se realize y de los medios tecnológicos utilizados.

El soporte del trabajo colaborativo puede requerir adaptar continuamente el sistema a las necesidades de los usuarios, los cuales pueden ir refinando sus requerimientos con el paso del tiempo y con la retroalimentación obtenida por el sistema.

Este proceso de continua evolución es uno de los mayores retos para el soporte efectivo del Awareness, el cual desde su concepción es dinánico y complejo.

5.2. Soporte del Awareness como requerimiento

El Awareness como requerimiento no es un tema muy tratado en la literatura científica. Como hemos constatado anteriormente, su suporte inicial se basa en guías, herramientas y mecanismos a nivel de diseño. Sin embargo, dado que es un aspecto crucial en los sistemas colaborativos, creemos que esta área requiere más trabajo de investigación y de desarrollo.

Los requerimientos de software son la piedra angular de cualquier desarrollo. El diseño de un sistema está basado en sus requerimientos. Sin ellos, no hay necesidad de implementar nada.

El Awareness se hace visible en las interfaces de usuario y en los diversos mecanismos interactivos que lo utilizan. Sin embargo, la representación del Awareness como requerimiento no está muy tratada actualmente en comparación con otros tipos de requerimentos de software.

Mica Endsley propone una técnica para descubrir los requerimientos de Awareness de Situación en [Endsley, 2001]. La importancia de esta técnica radica en que no se puede definir ningún requerimento si no se conoce. Esta técnica no forma parte del soporte del Awareness como requerimiento. Sin embargo, forma parte del paso previo: el análisis y la exploración de requisitos de información para la toma de decisiones y para la colaboración.

En el área del Groupware, [Teruel et al., 2011] ha extendido recientemente la metodología I* [Castro et al., 2001] para integrar el Awareness a la definición de requerimientos del sistema. La extensión para introducir requerimientos de sistemas CSCW contempla el Awareness como un nuevo concepto llamado *Awareness softgoal*, el cual representa una necesidad especial de percepción de la presencia o acciones de otros usuarios, sin la cual el desarrollo de la tarea sería difícil o no podría hacerse. En conjunción con dicho concepto se agrega el *Awareness resource*, el cual corresponde con la implementación o una solución en diseño para cumplir un objetivo *Awareness softgoal*.

El Awareness puede encontrarse a nivel de requerimiento en metodologías específicas para desarrollar Groupware, lo cual se describe en la Sección 5.4.

5.3. Soporte a nivel de interfaces de usuario

Ya que proveer información es una tarea muy relacionada a las interfaces de usuario, varios investigadores han desarrollado *toolkits* gráficos para agilizar la tarea de soportar los tipos de Awareness más comunes en un sistema Groupware.

Uno de los primeros, Groupkit [Roseman and Greenberg, 1996] incluyó *widgets*[5] para el soporte de varios aspectos del Workspace Awareness.

Los widgets soportados inicialmente por Groupkit proveen tres funcionalidades:

- **Estado del participante**: Provee información bajo demanda de cada participante en línea. Figura 27.
- **Telepuntero**: Utiliza un puntero especial para mostrar cual es la posición del puntero de cada participante en el espacio de trabajo compartido.
- **Awareness de localización**: Provee información sobre la localización relativa de cada participante en el sistema. Dependiendo del widget, puede usarse para saber en qué parte de un documento se encuentra o en qué parte de un mapa. Figura 28.

En la misma línea que Groupkit, MAUI [Hill and Gutwin, 2004] proporciona un toolkit gráfico para el desarrollo de interfaces colaborativas enfocadas a proporcionar Awareness de grupo.

Proporcionar Awareness para el soporte de la colaboración requiere incluir el elemento *usuario* o *grupo* a la información proporcionada. Además, dicha información se puede proporcionar síncrona o asíncrona.

[5] `http://en.wikipedia.org/wiki/GUI_widget`

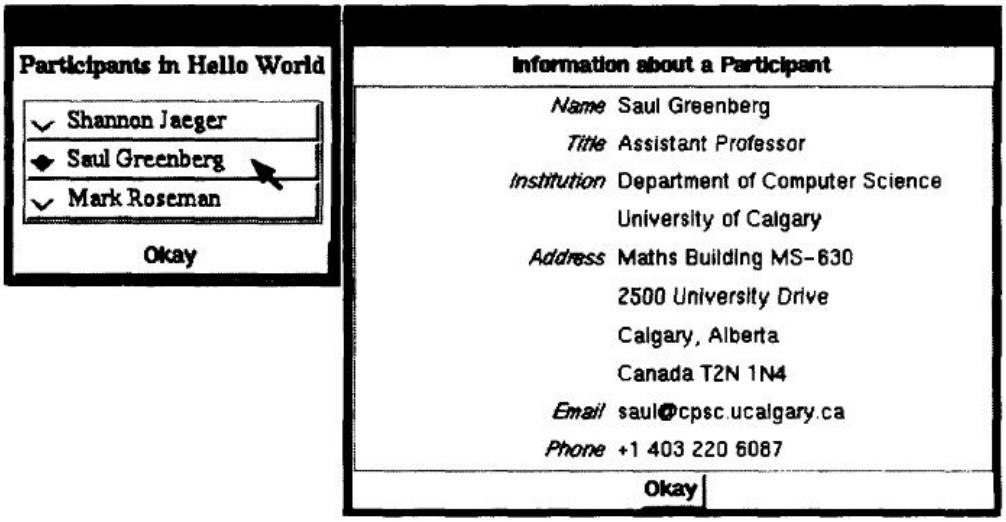


Figura 27. Widget sobre el estado de un participante.

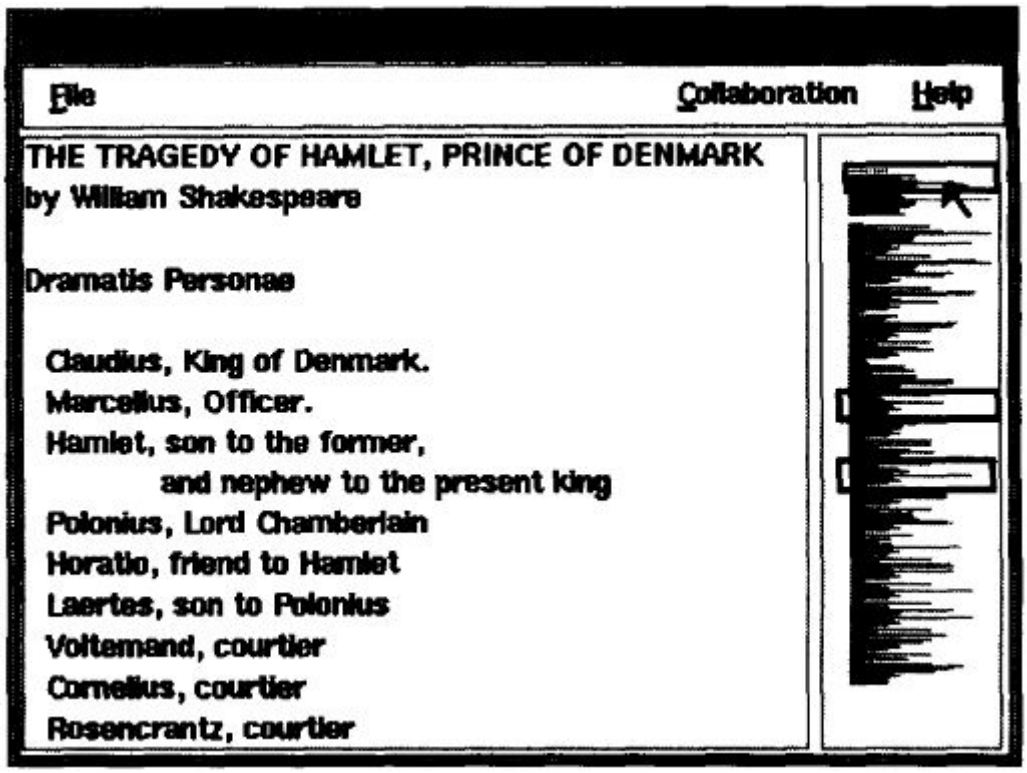


Figura 28. Widget sobre la locazación de los participantes durante la edición colaborativa de un documento.

MAUI utiliza un sistema de eventos para obtener y proporcionar los datos de Awareness. Es decir, eventos para capturar información y eventos para proporcionarla. Además el sistema de eventos les permite a los widgets gráficos ser utilizados en tareas interactivas.

MAUI incluye una serie de widgets gráficos comúnes adaptados al uso colaborativo, como son botones, menús, deslizadores, cuadros de selección, cuadros de texto, tabuladores, listas y tablas.

En la Sección 6 se presenta un estudio más extenso de las interfaces de usuario para el soporte del Awareness a nivel general, no solo a nivel colaborativo.

5.4. Soporte en metodologías de desarrollo

El soporte del Awareness en las metodologías de desarrollo es muy reciente. En la línea de los sistemas colaborativos, Penichet ha desarrollado una serie de plantillas para la extracción y definición de requerimentos de Awareness [Penichet et al., 2009a]. Estas plantillas están integradas a la metodología TOUCHÉ (Capítulo 2 Sección 3.2) y están orientadas la desarrollo de sistemas Groupware.

Esta metodología reconoce la importancia del Awareness y de su definición como requerimientos de información que deben ser cumplidos por los desarrolladores.

Para representar un requerimiento de Awareness usando el enfoque de [Penichet et al., 2009a], se anexa la siguiente información (*Awareness Issues* o *Necesidad de percepción*) por cada requerimiento de Awareness a la definición de una tarea, agrupando dicha información según los usuarios a los que va destinada:

- *¿Qué?*: Información que se requiere.
- *¿Cómo*: Forma de presentarla, ya sea como gráfica, mapa, widget colaborativo específico, etc.

- *¿Cuándo*: Presentar en tiempo real o no.
- *¿Dónde?*: Lugar en donde mostrar dicha información, como puede ser el mismo espacio de trabajo o una ventana distinta.
- *¿Por qué*: Justificación a nivel de usuario de porqué se requiere esta información.

En TOUCHÉ se maneja el Awareness como un objetivo o requerimiento de información ligado a alguna tarea o rol. Estas necesidades de conciencia se enfocan al *Worskpace Awareness* y se soportan a nivel de interfaz de usuario a través de las interfaces abstractas incluidas para este propśito.

No queda muy claro como TOUCHÉ mantiene la trazabilidad entre las interfaces de usuario abstractas específicas para el Awareness de Espacio de trabajo y los requerimientos en donde se especifican las Necesidades de conciencia. Además no pueden soportarse otros tipos de Awareness diferentes al Worskpace Awareness sin cambiar la especificación de requerimientos, los mecanismos de trazabilidad y los mecanismo de generación de interfaces de usuario.

AMENITIES (Capítulo 2 Sección 3.1) es otra metodología para el desarrollo de sistemas Groupware que de alguna manera soporta el Awareness en su proceso, aunque no directamente ni usando la misma terminología relacionada con el Awareness.

En AMENITIES el soporte del Awareness es a través de *artefactos* ligados a tareas, los cuales pueden tener como objetivo el transmitir *objetos de información*. Dichos *artefactos* pueden desplegarse a través de interfaces de usuario específicas para el Awareness, tal y como se muestra en las secciones 6.8 y 6.9.

Los *objetos de información* pueden representar desde documentos hasta mensajes y también información de Awareness, lo cual deja un poco al aire varias cuestiones sobre como trazar los mecanimos de Awareness o sobre quién puede y no puede accederla.

En el caso de la metodología CIAM (Capítulo 2 Sección 3.3)

La metodogía CIAM [Molina et al., 2009] ha integrado recientemente el soporte del Awareness [Gallego et al., 2011]. Esta extensión integra la definición de ciertos aspectos del Awareness a la definición de las tareas durante el desarrollo. Estos aspectos del Awareness se desarrollan hasta las interfaces de usuario colaborativas, para las cuales se integran widgets gráficos específicos.

El Awareness soportado por la metodología CIAM está basado en el Workspace Awareness y aunque no provee mecanismos a nivel de desarrollador para soportar otros tipos de Awareness, provee las bases para hacerlo a través de su metamodelo, lo cual requiere extender de nuevo la notación que soporta la metodología [Molina et al., 2006].

La extensión de CIAM propuesta para el soporte del Awareness está enfocada al Workspace Awareness, aunque dado su metamodelo puede extenderse para soportar otros. Las *widgets* a nivel de diseño también están orientadas al soporte del Workspace Awareness y no está clara su trazabilidad desde dichos componentes de la interfaz de usuario hasta las tareas que las utilizan. Tampoco se provee una forma para delimitar o restringir el tipo de información de Awareness proporcionada, por lo que queda a nivel de implementación el controlar dichos procesos.

6. Interfaces de Usuario para el Awareness

Las interfaces de usuario (UI) permiten la interacción entre un sistema software y sus usuarios, los cuales son mayormente humanos. Si los usuarios fueran agentes inteligentes no se usarán UIs, sino APIs de comunicación proporcionadas por el sistema.

Entre las diferentes funciones de una UI, se encuentra la de transmitir información. La información que nos interesa en este trabajo es la referente al Awareness.

Transmitir información de Awareness implica actualizar la UI y cambiar los valores que se están transmitiendo de forma directa o indirecta.

En las siguientes subsecciones presentaremos una guía para dar soporte a las interfaces de Awareness, las cuales no son otra cosa que interfaces de usuario especializadas en la transmisión de información cambiante usando todas sus características, sin dejar de ser usables para otros fines.

6.1. Interfaces multimodales y la expresion del cambio

Las UIs no están restringidas a un solo canal de interacción. Existe un grupo de UIs llamadas Interfaces de Usuario Multimodales [Tzovaras, 2008], las cuales permiten combinar UIs diseñadas para distintos canales de comunicación, de tal forma que pueden distribuir mejor los datos transmitidos a través de distintos canales de comunicación. Esta funcionalidad permite evitar la sobrecarga en los canales sensoriales o permite reafirmar cierta información de importancia usando dos canales o más (múltiples señales con la misma información).

Por ejemplo, cuando un telfono móvil detecta que hay una llamada entrante, avisa al usuario mediante varios mecanismos orientados a distintos sentidos: (1) varios sonidos o música son generados, (2) la interfaz gráfica cambia para avisar sobre la llamada y (3) el sistema de vibración del teléfono genera movimiento para estimular el tacto.

6.2. Proceso de retroalimentación

Todos los elementos de una UI tienen diversas características que son captadas por el canal sensorial al que van destinadas. Por ejemplo, las interfaces gráficas tienen forma, color, intensidad de brillo, tamaño y combinaciones de éstas. Las interfaces auditivas tienen tonos, volumen, ritmo y combinaciones.

El cambio se representa a través de redefinir en tiempo real algunas de estas características, de tal forma que la UI transmita información nueva a partir de estos cambios.

Los cambios de forma, color, contenido textual, tamañó, todos ellos implican un cambio en la información que se está transmitiendo. Estos cambios y sus combinaciones permiten transmitir la información de Awareness.

6.3. Visión

La visión es el sentido más explotado por los sistemas informáticos. El uso de la audición normalmente queda relegada a apoyo secundario de la visión.

Prácticamente todas las UIs conocidas son gráficas. Existen patrones de interacción bien documentados en esta área[6] y pueden llegar a ser muy complejos en funcionamiento y capacidad de transmitir información.

La visión es muy propensa a sobrecargarse con información irrelevante. Basta ver las páginas web actuales saturadas de publicidad y otros elementos los cuales no aportan funcionalidad o capacidad de interacción, sino simplemente propuestas de compra.

Debido a esta clara preferencia de las interfaces gráficas, la mayoría de los trabajos encontrados sobre interfaces para proporcionar Awareness están basados en el uso de la vision mediante repsentaciones graficas de la interfaz. Esta situación puede considerarse normal si se toma en cuenta que la principal interfaz de salida de datos del ordenador es la interfaz gráfica, la cual se muestra a través de una pantalla o monitor.

La visión humnaa puede captar combinaciones de forma, tamaño, color, iluminación, textura, movimiento, grupos de entidades, y si lo que se observa son rostros de otros seres humanos se pueden captar algunas emociones como la alegría, la tristeza, el miedo, entre otros. Sin duda alguna, para una persona sin discapacidad visual, la visión es tal vez el sentido sensorial más utilizado.

Tal vez una de las tareas más complicadas a la hora de diseñar interfaces gráficas de usuario es la de transmitir la mayor cantidad de información sin

[6] `http://www.welie.com/patterns/`

saturar la visión, a la vez de que se resalte lo más importante y se le quite atención a lo no tan importante.

En el caso del Awareness, las interfaces gráficas pueden aplicar cambios a cualquiera de las características mencionadas a fin de capatar la atención y transmitir nueva información. Sin embargo, si la interfaz gráfica está muy saturada va a resultar muy difícil enterarse de lo que está pasando.

6.4. Audición

La audición en las interfaces de usuario es un sentido secundario. Normalmente la audición se utiliza para llamar la atención a cierta parte de la interfaz principal o para avisar de algún evento.

En los sistemas de mensajería la interfaz auditiva se utiliza para avisar sobre nuevos mensajes o para avisar cuando alguien de interés entra en el sistema.

Las alarmas son un ejemplo claro del uso de la interfaz auditiva. Encontramos alarmas en radios, relojes, coches, ordenadores, entre otros. La finalidad de una alarma es avisar de un evento.

Pueden aprovecharse mejor las interfaces auditivas ya que las combinaciones de tono, melodía, ritmo y volumen pueden identificarse y relacionarse con eventos concretos. Por ejemplo, hoy en es habitual en los teléfonos móviles poder configurarse para reproducir un sonido determinado, ayudandándole al usuario a identificar al llamante sin tener que ver la pantalla del teléfono móvil.

Otro tipo de interfaz de usuario no tan elaborada es la que permite un diálogo directo entre personas o agentes a través de un canal auditivo, como es el caso de los sistemas de voz tipo Skype[7], Google Talk [8] entre otros, los cuales permiten realizar llamadas o conferencias entre varias personas, algunas de ellas incluyendo la interfaz de video.

Los videojuegos también hacen uso exhaustivo de la interfaz auditiva trans-

[7] `www.skype.com`
[8] `http://www.google.com/talk/`

mitiendo sonidos que le permitan al jugador adentrarse en el mundo virtual generado por el propio juego. Por ejemplo, si en un juego aparecen lobos, un sonido común puede ser el aullido de un lobo.

Hay que tener cuidado con el uso de las interfaces auditivas, ya que no todos los usuarios responden igual a los estímulos sonoros. El volumen y el ritmo pueden generar mucho estrés si se utilizan de manera indebida, como es el caso de varios *banners* de publicidad que pueden verse mucho por internet.

Podemos afirmar que este tipo de interfaz se satura más rápidamente que la interfaz visual y por tanto debe usarse con más cuidado.

6.5. Tacto

La interfaz táctil más conocida (o usada) por el ser humano común no está relacionada directamente con el ordenador, sino con los teléfonos móviles.

El llamado "*vibrador*" se utiliza para avisar de algún evento importante, como es una llamada entrante o una alarma. Su importancia radica en que normalmente no se tiene a la vista la pantalla del teléfono móvil o el entorno puede ser demasiado ruidoso como para que el usuario reciba las señales auditivas.

El vibrador funciona con un pequeño motor y tiene un gasto considerable de energía. Al transmitir movimiento lo puede hacer en distintos ritmos y con distintas intensidades y duraciones, con lo cual tiene un abanico considerable de señales que puede transmitir. Dada su cercanía con la piel, es la interfaz táctil más adecuada cuando no se puede observar ni escuchar el teléfono móvil.

Su uso ha sido probado en conjunción con sistemas informáticos para dar notificaciones [Riener and Ferscha, 2008, Sahami et al., 2008].

Una interfaz táctil menos utilizada y específica para un grupo de usuario en particular es la interfaz para leer el alfabeto Braille [9]. Dichas interfaces utilizan sistemas eletro mecánicos para desplejar caracteres del alfabeto de forma que

[9] `http://en.wikipedia.org/wiki/Braille`

Figura 29. Pantalla tipo Braille para desplegar caracteres en alfabeto Braille de forma táctil.

pueda tocarse por el usuario (Figura 29 [10]).

El uso de las interfaces de usuario táctiles no es generalizado. De hecho, podemos afirmar que es casi nulo. Puede ser la falta de dispositivos, la falta de APIs adecuadas, u otras razones. Pero ciertamente el sentido del tacto puede aprovecharse para transmitir Awareness y hoy en día es un área que debe explorarse a fin de reducir la carga de información sobre otros canales sensoriales, como son la vista y la audición.

6.6. Otras capacidades sensoriales

Actualmente no existe un uso generalizado de interfaces de usuario que utilicen sentidos alternativos a la vista y a la audición. Como vimos anteriormente, es difícil encontrar un uso común de las interfaces de usuario táctiles con capacidad de transmitir información, como es el caso del vibrador de los teléfonos móviles.

El cuerpo humano es capaz de sentir calor o frio (temperatura), presión atmosférica, electricidad, ciertas conexiones mentales como el enigmático *sexto sentido*. Estos sentidos son prácticamente ignorados en las interfaces de usuario actuales, ya que requieren mucho trabajo técnico para proporcionar de forma segura señales a través de cualquiera de estos sentidos.

El gusto y el olfato tampoco son utilizados en las interfaces de usuario, principalmente por la incapacidad de generar señales adecuadas para ello o por lo complicado que sería recibirlas en el caso del gusto.

Esta área sigue abierta a la investigación y su uso para proporcionar Awa-

[10] `http://www.deafblind.com/display.html`

reness puede ser interesante aunque fuera de lo común. Se siguen aplicando a estos canales las técnicas para transmitir Awareness: **aprovechar el cambio en cada una de las característcicas de las señales enviables por dicho canal**.

6.7. Interfaces de usuario para sistemas de monitorización

Existen diversas formas de monitorización dependiendo de los objetivos y de la tecnología disponible.

Para transmitir datos cambiantes se pueden utilizar los distintos atributos de las interfaces de usuario que son usadas para transmitir los datos. Si los cambios son importantes es necesario llamar la atención de los usuarios usando formas distintas de mostrar los datos o utilizando otras interfaces de apoyo.

Por ejemplo, cuando un valor se sale de un rango, el sistema puede producir sonidos, utilizar colores llamativos como el rojo o el verde, utilizar el movimiento o el parpadeo de la interfaz, etc. Estas formas de llamar la atención son fácilmente reconocibles en coches, páneles de control, electrodomésticos, etc.

Otra forma de monitorizar es la de tener acceso a todos o a una parte de los datos que han cambiado. Por ejemplo, los sistemas bitácora o *logs* de acceso, los cuales mantienen la bitácora de todos los cambios de interés para esa tarea. Por ejemplo, los sistemas IRC[11] comúnmente mantienen todos los mensajes públicos en una sola sección de la interfaz, los cuales son enviados a todos los usuarios. De esta forma, todos los usuarios pueden ver todos los mensajes públicos que hay y que se han escrito (con ciertas limitantes técnicas). Figura 30[12].

Una forma común y menos intrusiva de monitoreo es la obtención de información bajo demanda. Es decir, el usuario no recibe constantemente los datos de la entidad que monitoriza, sino que recibe solamente un indicador de que alguno de los valores de interés ha cambiado. Es entonces cuando el usuario de-

[11] `http://en.wikipedia.org/wiki/Internet_Relay_Chat`
[12] Imagen obtenida de `http://sourceforge.net/projects/ychat-irc/`.

Figura 30. Cliente IRC en el que se muestra la ventana principal de mensajes públicos visibles y para todos los usuarios. También se mantiene el historial de mensajes y eventos de interés como entradas y salidas de usuarios. Normalmente los eventos se enlazan con el tiempo en el que sucedieron, aunque en este cliente en particular no sea así.

cide si ignorar dicho aviso o inspecciona a detalle la instancia que ha cambiado. Las ventanas de información adicional son un ejemplo típico de esta situación. Figura 31.

Estas tres técnicas básicas de comunicación pueden combinarse y agruparse para mostrar toda la información de Awareness que se necesite. Las dos primeras técnicas son pasivas y la última es activa.

Las técnicas pasivas pueden saturar al usuario o distraerlo de una tarea importante. Las técnicas activas de obtención de datos normalmente requieren que el usuario haga algo y por lo tanto, también lo distraen. La combinación de estas dos técnicas puede equilibrar estas ventajas y desventajas.

6.8. Interfaces de usuario para sistemas interactivos

Los sistemas puramente informativos prácticamente no proporcionan medios para la interacción del usuario con el sistema, salvo las puramente necesarias. Por otra parte, los sistemas de gestión, ocio, juegos, entre otros proveen diversas técnicas para que los usuarios manipulen el sistema y puedan realizar las tareas

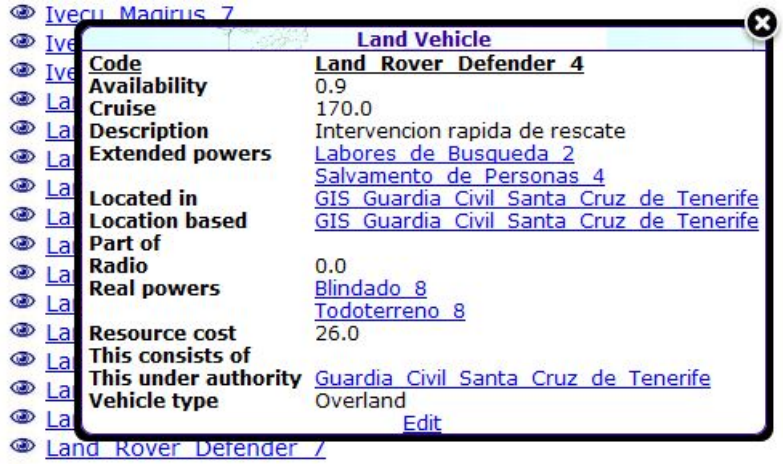


Figura 31. Ventana informativa desplegada bajo demanda.

correspondientes para cumplir sus objetivos.

Las técnicas de interacción combinan el uso de interfaces de usuario informativas o de Awareness con el uso de controles (botones, listas, cuadros de selección, etc.) o interfaces de navegación (hiperenlaces, menús, etc.).

Por ejemplo, un cuadro de texto puede integrar un comportamiento para transmitir al usuario que los datos introducidos son correctos o incorrectos. Podemos llamarle Awareness de validez, aunque probablemente nadie lo haya relacionado nunca con el Awareness (no porque no esté relacionado, sino por la falta de información al respecto).

De hecho, prácticamente todos los widgets para control implementan de alguna manera mecanismos de Awareness para transmitir su estado al usuario. Esto es lo que en sistemas interactivos se conoce como "retroalimentacion" o *feadback*, lo cual tiene una relación directa con el Awareness. Por ejemplo, un boton muestra su estado de deshabilitado cambiando a un tono grisáceo y dejando de mostrar el icono característico cuando se está encima del mismo. Un cuadro de texto puede opacarse e impedir que se introduzcan datos en él. Figura 32[13].

[13]Imagen obtenida de `http://www.codeproject.com/Articles/9952/ComboBox-with-read-only-behavior`.

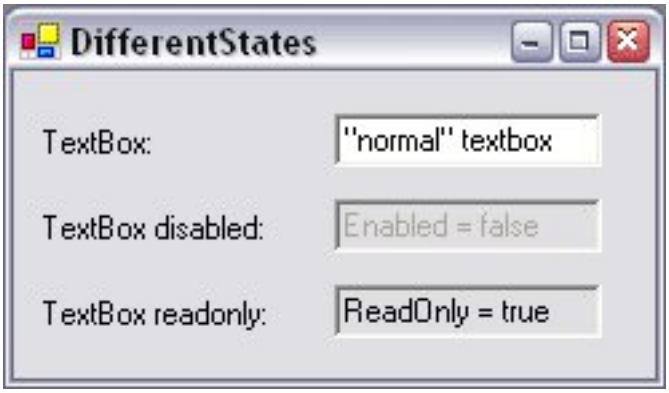


Figura 32. Cuadro de texto mostrando diferentes características visuales según su estado. El cuadro de texto proporciona Awareness de su estado de escritura de manera visual.

El estado de los controles debe transmitirse para proporcionar una retroalimentación al usuario, ¿Pero porqué la necesidad de esa retroalimentación? **Porque se necesita Awareness del estado de los controles para poder interactuar con ellos y con el sistema a través de ellos**.

6.9. Interfaces de usuario para sistemas colaborativos

Las interfaces de usuario para los sistemas colaborativos normalmente agregan el componente *Usuario*, *Grupo* o *Rol* a toda la información que transmiten. De esta manera se diferencia la información de un usuario con respecto a la de otros.

Una UI de Awareness transmite datos cambiantes e importantes para el usuario. En un entorno Groupware la importancia de la información recide en su importancia para cada individuo y en la importancia en conjunto para todo el grupo.

La diferencia entre los conceptos de cooperación y colaboración implica también una diferencia entre la información de Awareness relevante en cada uno de estos procesos, así como en las actividades que se realizan en ellos.

Por ejemplo, en una tarea cooperativa dividida en varias tareas individuales puede ser importante saber quien desarrolla cada tarea, ya que estas responsa-

Figura 33. Ventana que muestra el estado de los usuarios en un sistema chat. De cada usuario se muestra su nombre de usuario, avatar, estado de disponibilidad.

bilidades pueden ser dinámicas y quien realiza una tarea ahora puede hacer otra la siguiente vez.

En una tarea colaborativa puede ser más importante saber quienes están realizando una tarea en conjunto o saber quienes trabajan bien juntos.

Aunque los propósitos para cada tipo de Awareness pueden variar dependiendo del dominio y del contexto, muchas interfaces de usuario serán muy similares, principalmente en la inclusión del usuario y su contexto individual y grupal como parte de la información.

A continuación presentamos interfaces de usuario utilizadas para tareas comunes en los entornos Groupware.

Para mostrar el estado de los usuarios conectados o disponibles se utilizan interfaces como la de la Figura 33[14]. Dichos elementos informativos se utilizan para proporcionar el Awareness de presencia o disponibilidad. Además proporciona otros datos como el avatar. Normalmente estas interfaces permiten obtener información de los usuarios bajo demanda, con lo cual se mejora el Awareness acerca de los usuarios.

Una forma de mostrar las actividades de un equipo de trabajo es a través de

[14]Imagen obtenida de `http://adium.im/screenshots/`.

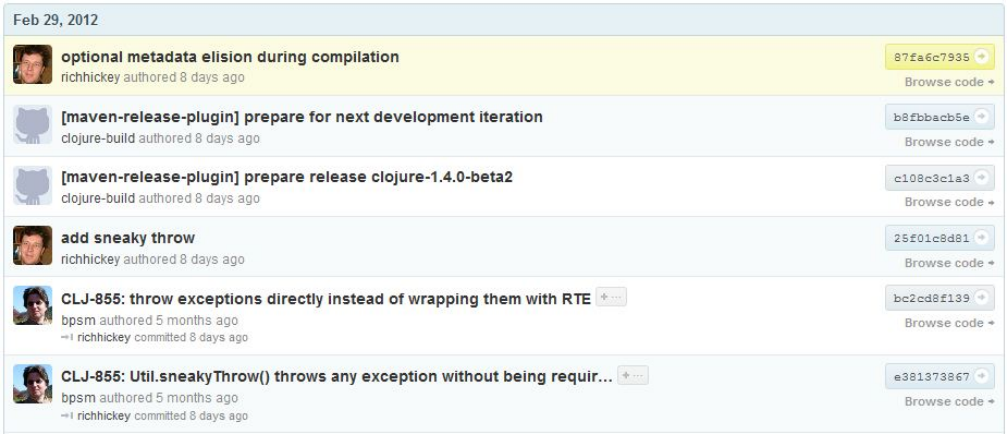


Figura 34. UI que muestra la última activdad (*commits*) en un repositorio de código.

una interfaz que muestre la información principal (dependiendo de las tareas) de las actividades realizadas así como la información necesaria de los autores. Figura 34[15].

Las interfaces tipo bitácora permiten generar un Awareness más avanzado, como es el caso del nivel de proyección, el cual depende en gran medida de los datos antigüos. Por otra parte, estas interfaces ocupan mucho espacio y solo son utilizadas para darse una idea global de en qué se está trabajando y que se ha hecho a lo largo del tiempo.

Las aplicaciones síncronas implican mucha transferencia de información entre los usuarios y el sistema, lo cual hoy en día es una de las principales dificultades técnicas para soportar Awareness, aunque no es la única.

La transferencia de información en tiempo real entre los usuarios de un grupo se requiere principalmente en las tareas colaborativas en tiempo real; es decir, todos a la vez.

Varias interfaces de usuario especializadas para dar soporte al Awareness en este tipo de aplicaciones se muestran a continuación. Varias de ellas desarrolladas principalmente para proporcionar Awareness de espacio de trabajo:

[15]Imagen obtenida de `https://github.com/clojure/clojure/commits/master`.

Figura 35. Vista de radar en un espacio de trabajo compartido. La localización y alcance de visión están representados con cuadros punteados o sombreados. La identidad de los usuarios se proporciona por una imagen.

La vista radar se utiliza para conocer la localización de los usuarios en un espacio de trabajo compartido, así como para saber que tanto pueden ver. Figura 35.

Una UI similar y mucho más utilizada es la vista de geolocalización. Pueden utilizarse mapas geográficos (Figura 36[16]), urbanos o virtuales (Figura 37[17]).

[16]Imagen obtenida de `http://www.openbsd.org/images/map.jpg`.
[17]Imagen obtenida de `http://curiosity-media.discovery.com/mediaItems/4/e/7/4e7353c19cdb3/second-life-3.jpg.jpg?v=1316180935`.

Figura 36. Localización geográfica de algunos de los desarrolladores del proyecto OpenBSD `http:www.openbsd.org`. La localización no es exacta pero la intención es los estados y paises en donde viven.

Figura 37. Sala de baile de la red social Second Life. En los sistemas 3D la localización se proporciona mediante los propios avatares de los usuarios dentro del entorno 3D.

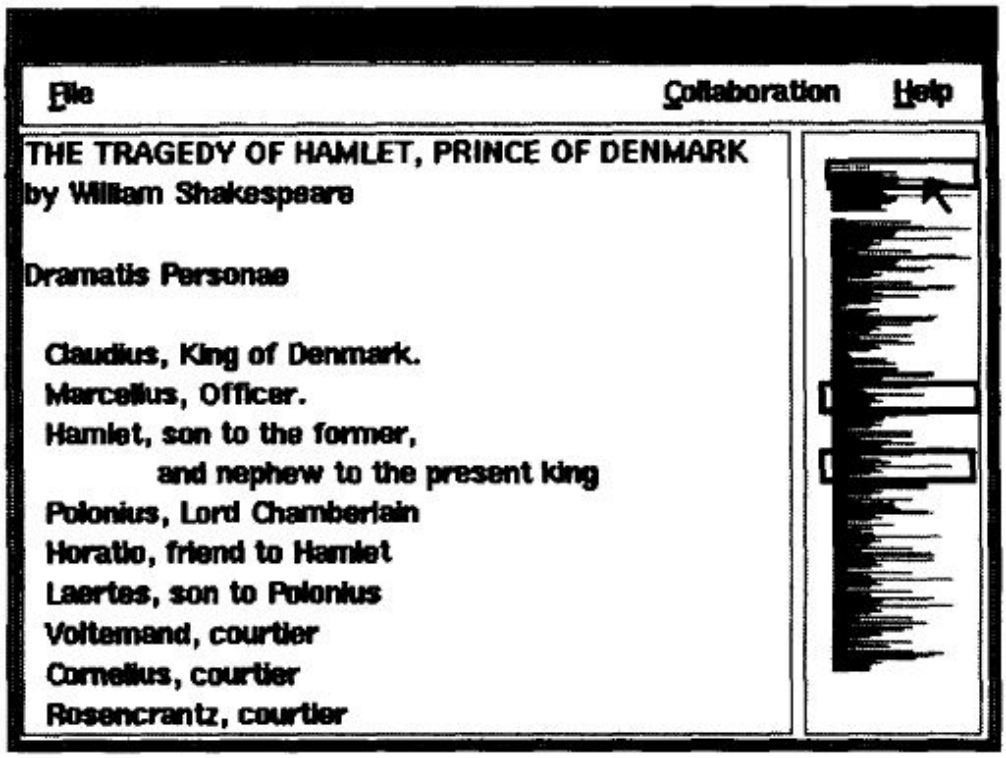


Figura 38. Vista Gestalt del toolkit GroupKit[Roseman and Greenberg, 1996] en el que se muestra la localización y el alcance de visión de los colaboradores durante la edición colaborativa de un documento.

La vista Gestalt [Roseman and Greenberg, 1996] transmite la localización de los usuarios dentro de un espacio de trabajo colaborativo muy particular: un texto compartido. Figura 38.

Las barras de desplazamiento colaborativas [Hill and Gutwin, 2004] también proporcionan Awareness de localización sobre un espacio de trabajo compartido de tamaño superior a los monitores utilizados.

En entornos de aprendizaje colaborativo no solo importa el espacio de trabajo compartido, sino también el conocimiento compartido (Sección 4.6) . El *árbol del conocimiento*[Kwon et al., 2011] (Figura 39) es una manera de representar el conocimiento compartido general de un grupo de individuos, de tal forma que es posible saber quién sabe más de algún tema en particular o quiénes tienen conocimientos de un tema. Hay que diferenciar entre el conocimiento

Figura 39. La representación del *Árbol del conocimiento*, el cual genera nuevas ramas y nuevas hojas según vayan aumentando las interacciones académicas y el conocimiento.

que solo posee un individuo pero que se sabe que dicho individuo lo tiene, y el conocimiento realmente compartido, que es el que todos tienen.

El diseñar interfaces de usuario no es una ciencia exacta. Algunas interfaces no son aptas para todo el mundo o para un uso específico. Algunos trabajos referentes a las interfaces de usuario para el Awareness se centran en guiar el diseño de estas interfaces dependiendo de su propósito y del Awareness que pretenden ofrecer. Por ejemplo, [Carroll et al., 2003] ofrece una guía extensiva para diseñar interfaces de usuario para el Awareness de Actividad, el Awareness de Acción y el Awareness Social o de Grupo, evaluando factores situacionales, grupales, de tarea y de herramientas.

Con un propósito diferente, [Kim and Kim, 2007] presenta un conjunto de guías para diseãr interfaces de usuario que tomen en cuenta la privacidad a la hora de proporcionar varios tipos de Awareness.

Las interfaces de usuario de Awareness para el Groupware son propensas a sobrecargar los canales de transmisión debido a la gran cantidad de información que deben transmitir (el Awareness exclusivo para el usuario y el Awareness sobre los otros usuarios). Un gran esfuerzo de diseño es necesario para elaborar dichas interfaces de la forma más conveniente a la situación, los objetivos y los usuarios.

7. Dificultades derivadas del soporte del Awareness

La formación del Awareness no es un proceso lineal y está sujeto a una gran cantidad de variables, ya sea de la persona o agente que crea el Awareness, como del entorno o del sistema el cual le proporciona activa o pasivamente la información necesaria para crear el Awareness.

Todas las dificultades inherentes a la generación del Awareness están presentes en los sistemas informáticos que buscan darle soporte, ya que son intermediarios entre el entorno y los actores que requieren Awareness para realizar cualquier

tarea.

Según el estudio realizado por Mica Endsley, existen dos grupos de factores que afectan la formación del Awareness en los sistemas dinámicos [Endsley, 1995]:

1. Los factores referentes a los propios seres humanos.
2. Los factores referentes al sistema y a la tarea a realizar.

En esta sección nos adentramos en varias de estas dificultades, la cuales afectan de manera significativa la adquisición de Awareness por parte de los actores y al soporte del Awareness por parte de los sistemas software que proporcionan la información del entorno de interés.

En relación con los aspectos de los seres humanos que intervienen en la formación del Awareness tenemos la pre-atención, la atención, la percepción, la memoria de trabajo y los objetivos. Otros factores como el estrés y la carga cognitiva también están ligados al propio ser humano. Sin embargo, en el sentido del Awareness, están más ligados al diseño del sistema y a las tareas, lo cual se trata en la Sección 7.2.

Las conclusiones sobre esta sección se tratan al final de la misma, ya que el Awareness no solo depende del sistema, sino también del entorno de trabajo y de los propios usuarios.

7.1. Referentes a los seres humanos

Los aspectos humanos que afectan la formación del Awareness están un poco desligados del sistema. Sin embargo, requieren tratarse con cuidado y preparar el sistema para apoyar estos aspectos, como son la pre-atención y la atención, o la memoria de trabajo.

Estos aspectos tienen que ver con las características mentales y sociales de cada individuo y por tanto, son un tanto únicas para cada uno de ellos. A continuación explicamos con más detalle algunas de ellas.

7.1.1. *Pre-atención*

De acuerdo con [Wickens, 1992], las características del entorno son inicialmente procesadas en paralelo a través de "preattentive sensory stores" en los cuales ciertas propiedades son detectadas, como es la proximidad espacial, el color, las propiedades simples de una forma, o el movimiento [Treisman and Paterson, 1984], proporcionando señales de una atención más localizada.

Estos objetos que son más relevantes, basados en la características registradas por la pre-atención, serán procesadas posteriormente usando una atención focalizada para alcanzar la percepción.

Las señales relevantes, entonces, tendrán un importante impacto en las porciones del entorno que sean inicialmente percibidas, y estos elementos formarán las bases para el primer nivel del Awareness según [Endsley, 1995]: la percepción.

7.1.2. *Atención*

El desarrollo de la atención en el proceso de percepción presenta ciertas restricciones a la habilidad de una persona para percibir correctamente múltiples elementos en paralelo, y como tal, es el mayor límite para la generación del Awareness.

La atención directa es necesaria para percibir y procesar las señales atendidas. Además, es nesearia para la toma de decisiones y y ejecución de acciones.

En entornos dinámicos y complejos, la demanda de atención resultado de la sobrecarga de información, toma de decisiones complejas y la ejecución de múltiples tareas pueden fácil y rápidamente exceder los límites de la capicidad de atención de un ser humano.

En un estudio sobre el Situation Awareness en pilotos de aviones, Fracker [Fracker, 1989] mostró que una limitada cantidad de atención es ocupada para elementos del entorno basada en su habilidad para contribuir al éxito de la tarea.

Ya que la cantidad de atención es limitada, un incremento en la atención a ciertos elementos (y por lo tanto, al Awareness de dichos elementos) significa la pérdida de atención (y de Awareness) sobre otros elementos del entorno una vez que el límite de atención es alcanzado, lo cual puede ocurrir muy rápidamente en entornos complejos.

7.1.3. *Percepción*

Además de afectar la selección de elementos para la percepción, la forma en la que la información es percibida es dirigida por el contenido de la memoria de trabajo y la memoria de largo plazo.

Un conocimiento avanzado de las características, forma y localización de la información, por consiguiente, puede facilitar significativamente la percepción de la información [Humphreys, 1981].

7.1.4. *Memoria de trabajo*

Una vez percibida, la información es almacenada en la memoria de trabajo. En la ausencia de otros mecanismos (como la memoria de largo plazo), la mayoría del procesamiento activo de información en las personas ocurre en la memoria de trabajo.

La nueva información debe combinarse con el conocimiento existente y debe desarrollarse una imagen global de la situación. Las proyecciones de un estado futuro y las decisiones posteriores sobre el curso apropiado de acciones deben ocurrir también en la memoria de trabajo. En estas circunstancias, una carga muy fuerte es impuesta sobre la memoria de trabajo, lo cual se agrava al alcanzar niveles superiores de Awareness (Comprensión y Projección), al formular y seleccionar respuestas y al ejecutar las acciones posteriores.

Wickens [Wickens, 1984] ha señalado que las predicciones de estados futuros (proyecciones de Awareness) imponen una carga muy pesada en la memoria de trabajo al requerir el mantenimiento de las condiciones presentes, condiciones

futuras, reglas usadas para generar las últimas a partir de las primeras y las acciones que son apropiadas para las condiciones futuras.

7.1.5. *Memoria de largo plazo*

En la práctica, las estructuras de la memoria de largo plazo pueden usarse para eludir o reducir las limitaciones de la memoria de trabajo.

Algunas de las caracterizaciones sobre como se organiza el conocimiento en el cerebro son los esquemas y los modelos mentales, los cuales se han detectado como agentes importantes para la toma de decisiones efectiva [Braune and Trollip, 1982] y que juegan un papel importante para el Awareness.

Los *esquemas* proporcionan marcos de trabajo coherentes para entender la información, comprendiendo componentes de sistemas altamente complejos, estados y funcionamiento [Mayer, 1983]. Muchos de los detalles de la situación se pierden cuando la información es codificada de esta manera, pero la información se vuelve más coherente y organizada para ser almacenada, recuperada y procesada posteriormente. Un solo esquema puede servir para organizar varios conjuntos de información y como tal tendrá variables que pueden llenarse con las particularidades del caso considerado. Un guión o "*script*" (un tipo especial de esquema) provee secuencias de acciones apropiadas para diferentes tipos de ejecución de tareas.

Los lazos entre los esquemas y los guiones pueden facilitar el proceso cognitivo porque un individuo no tiene que decidir activamente las acciones apropiadas en cada momento sino que lo sabrá automáticamente basado en uno de estos guiónes.

Un modelo mental [Rouse and Morris, 1985] es un mecanismo con el cual los humanos son capaces de generar descripciones del propósito y de la forma de un sistema, explicaciones del funcionamiento del sistema y de sus estados observados y predicciones de su estado futuro. Pueden describirse como esquemas complejos que son usados para describir el comportamiento de los sistemas. Por lo tanto,

un modelo mental puede verse como un esquema para cierto sistema.

Estos modelos mentales se desarrollan al observar constantemente sistemas complejos y dinámicos. Mientras más complejo y dinámico es el sistema, más tiempo, capacidades y experiencia se requerirán para generar los modelos mentales que esquematicen dicho sistema.

Estos modelos mentales permiten tomar decisiones más efectivas y rápidas, ya que el ser humano puede valerse del conocimiento del sistema y su comportamiento, así como de los guiones que han resultado ser más efectivos en dichas situaciones. Para lograr esta capacidad no hay otro camino que la experiencia y el entrenamiento.

7.1.6. *Objetivos*

El Awareness no es generalmente manejado como algo que existe únicamente para sí mismo. El Awareness es necesario para la toma de decisiones relacionadas con alguna tarea o sistema. Como tal, está íntegramente ligado con el contexto y las decisiones para las cuales el Awareness es buscado. Es decir, **está fundamentalmente ligado a los objetivos de la persona.**

Los objetivos forman las bases para la toma de decisiones en entornos diñamicos. Además, puede haber más de un objetivo simultáneo, y a veces pueden entrar en conflicto. En la mayoría de los sistemas, las personas no son entes incapaces de otra cosa que no sea recibir datos del entorno, por el contrario, son activos buscadores de datos en luz de sus propios objetivos.

Según los objetivos y las expectativas de cada usuario, su atención será dirigida a los elementos de Awareness que según su experiencia (esquemas y modelos mentales) le permitan cumplir sus objetivos.

7.2. Referentes al sistema y a las tareas

Hemos revisado varios de los aspectos humanos que afectan la formación del Awareness y que pueden dificultar su soporte en un sistema software. Ahora bien,

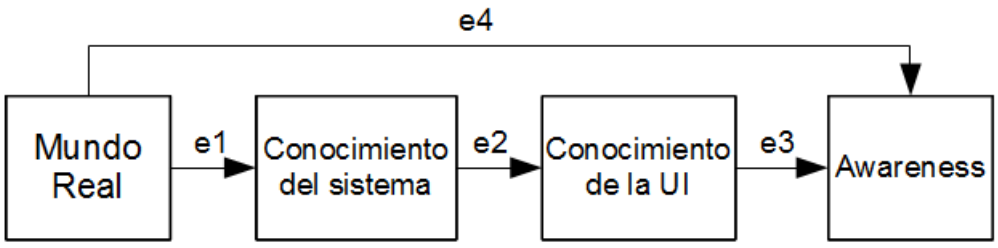


Figura 40. Entradas de información desde el mundo real hasta llegar a la formación del Awareness en un individuo

existen varios aspectos relacionados con el propio sistema y con las tareas que se pueden ejecutar sobre él para cumplir los objetivos del usuario.

Como puede verse en la Figura 40, entre el Awareness en los usuarios y el "*mundo real*" o virtual se encuentra el sistema software y sus interfaces de usuario, los cuales transmiten la información del mundo real a los usuarios. Las entradas están denotadas por la letra "*e*" y van desdel mundo real al sistema (*e1*), luego a las interfaces de usuario (*e2*) y luego al usuario, que es quien forma el Awareness (*e3*). La entrada *e4* va del mundo real a los usuarios y siempre lleva más información que la entrada *e3*, con todo lo que ello representa.

Ya que el sistema es un intermediario, su diseño interviene mucho en el Awareness final que el usuario puede obtener de la información proporcionada por el sistema. Dicha información puede ser la misma que la percibida por el sistema, pero definitivamente el mayor cuello de botella está en las interfaces de usuario, las cuales están muy orientadas al canal visual, dejando de lado prácticamente todos los demás sentidos del ser humano.

7.2.1. *Diseño del sistema*

En lo que respecta al Awareness, el sistema debe proporcionar los datos del entorno de trabajo o "*mundo real*" a los usuarios. Lo ideal es que esos datos se transmitan por los mismos canales que son recibidos, pero eso no es posible por la falta de infraestructura adecuada.

Ya que el sistema puede procesar dicha información antes de transmitirla al usuario, puede enriquecerla y ayudar a los usuarios a alcanzar los niveles superiores del Awareness a través de dichos datos procesados previamente por el sistema. De este modo, a falta de poder transmitir todos los datos recibidos por los canales más adecuados, puede incrementar la cantidad de información en los otros canales y reducir la complejidad de los mismos, a fin de no afectar negativamente a los usuarios con más complejidad (Sección 7.2.7) y sobrecarga de datos (Sección 7.2.6).

7.2.2. *Soporte*

Una parte muy importante y carente en casi todas las metodologías de desarrollo es el manejo del Awareness.

A la hora de soportar el Awareness uno se pregunta cuál es el que hay que soportar, cómo definirlo en el sistema, como darle soporte a lo largo de todo el desarrollo, como transmitirlo a los usuarios.

Como vimos en el Capítulo 2 Sección 3, algunas metodologías integran el Awareness en varias de sus etapas. Sin embargo, el aspecto del modelado sigue estando pendiente. Por esa razón nuestro trabajo se centra en este aspecto del soporte del Awareness.

Los cambios en el software son necesarios y prácticamente inevitables. El mantenimiento de los componentes actuales y la integración de nuevos requerimientos puede implicar mucho trabajo de adaptación en el software. Ya que los requerimientos de Awareness están en constante evolución y experimientación, es de esperarse que el software cambie en ese aspecto. Sin los medios adecuados para soportar el Awareness, esta tarea puede crear más problemas que beneficios.

El soporte básico del Awareness implica identificarlo, modelarlo en el sistema, capturar los datos, tal vez procesarlos y proporcionárselos a los usuarios en las condiciones adecuadas y en el momento adecuado. Cada uno de estos pasos debe estar bien definidos y manejados para que no tener problemas durante el soporte

el mantenimiento del Awareness en un sistema.

7.2.3. *Diseño de la interfaz*

El propósito de las interfaces de usuario es el de permitir la interacción con el sistema. En el caso del Awareness su propósito principal es el transmitir información.

El diseño de las interfaces de usuario depende mucho del público objetivo, sus capacidades o gustos, y su entrenamiento. Una mala interfaz puede causar muchos problemas, como puede ser la poca expresividad de datos, la exigencia de demasiada atención, la sobrecarga de información, entre otros.

Un estudio más extenso sobre las interfaces de usuario para el Awareness se presenta en la Sección 6.

Dentro del grupo de usuarios objetivo siempre se pueden encontrar diferentes reacciones a una misma UI. Es decir, no todos los usuarios responden de la misma manera a una interfaz de usuario o a sus medios de transmisión de información. Algunas personas pueden ser más sensibles a ciertos aspectos de la interfaz y generar estrés o sobrecarga cognitiva en situaciones en las que otros usuarios no tienen ningún problema.

En estas situaciones los sistemas software pueden dotarse con mecanismos de control que permitan regular los aspectos de la UI que pueden ser problemáticos para los usuarios, como es el caso de los mecanismos para proporcionar información de Awareness.

Podemos destacar dos formas de control de la interfaz de usuario: la adaptatividad y la personalización [Benyon and Murray, 1993].

La adaptatividad se refiere a los mecanismos implementados por el sistema para cambiar aspectos de la interfaz de acuerdo a ciertas características de los usuarios. En este caso el usuario no tiene control sobre los cambios, ya que son manejados por el sistema.

La personalización se refiere a que el usuario pueda cambiar aspectos de su interfaz a través de controles establecidos previamente, de tal forma que él mismo cambie los aspectos que le resultan incómodos o dañinos.

Claramente la adaptación y la personalización pueden combinarse. Por ejemplo, los usuarios pueden activar o desactivar adaptaciones hechas por el sistema si ellos consideran que no son útiles.

Como mecanismos de control de los mecanismos de Awareness podemos encontrar los siguientes ejemplos:

- El filtrado de datos: Permite reducir la cantidad de información que se le transmite al usuario [David and Borges, 2001, Kirsch-Pinheiro et al., 2005].
- La desactivación de UIs de Awareness: Permite detener una parte del flujo de datos que se le envía al usuario [Morán et al., 2001]. Puede ser contra producente si el usuario realmente necesita esa información. La capacidad de desactivar interfaces debe manejarse con cautela y preferentemente deben proporcionarse otras señales para alertar sobre esta carencia de datos o mecanismos que impidan desactivar las UIs de Awareness si son vitales para los objetivos.

7.2.4. *Privacidad*

En el caso del Groupware la privacidad juega un papel muy importante, ya que se está trabajando con información de personas, las cuales pueden tener diferentes opiniones sobre la cantidad de información personal que pueden acceder los demás usuarios del sistema.

El manejo de la privacidad es complejo y requiere tenerlo en cuenta durante el desarrollo de cualquier sistema. Existen también trabajos que investigan directamente con los usuarios distintas técnicas sobre el manejo de la privacidad, con las presentadas por [Kim and Kim, 2007] y enfocadas a varios tipos de Awa-

reness en concreto, como son el Awareness de Presencia, el Awareness de Tarea, entre otros.

El enfoque tomado para nuestro trabajo se basa en el marco de trabajo presentado por Drury [Drury and Williams, 2002], cuyo manejo de la privacidad se basa en proporcionar la información de Awareness solamente al usuario que la requiera, cuando la requiera y de la forma que la requiera.

7.2.5. *Estrés*

El estrés es un factor dinámico que puede afectar considerablemente la generación del Awareness. Podemos encontrar varios factores que pueden provocar estrés durante la realización de una tarea [Sharit and Salvendy, 1982]:

- *Factores físicos*: Ruido, vibración, frío/calor, luz/oscuridad, condiciones atmosféricas, medicamentos, cansancio o fatiga, entre otros.
- *Factores psicológicos y sociales*: Miedo, ansiedad, incertidumbre, importancia o consecuencias, aspectos económicos, auto estima, pretigio, pérdida o ganancia de trabajo, sobrecarga mental, presión temporal, entre otros.

Varios de estos factores solo provocan estrés si el usuario lo permite o si no sabe como afrontarlos correctamente.

Cierta cantidad de estrés puede ser benificiosa ya que incrementa la atención a los aspectos importantes de la situación. Sin embargo, una cantidad considerable de estrés puede tener consecuencias negativas [Hockey, 1970].

Una consecuencia negativa del estrés es que las personas tienden a reducir su campo de atención para incluir únicamente ciertos aspectos centrales, dejando fuera otros que pueden ser decisivos para cumplir sus objetivos.

Otra consecuencia negativa del estrés es que reduce la capacidad de la memoria de trabajo y la capacidad de recuperar conocimiento [Hockey, 1986]. Si el Awareness depende en gran medida de la memoria de trabajo (por ejemplo,

para los niveles de comprensión y proyección), su generación se verá considerablemente afectada por el estrés.

7.2.6. *Sobrecarga cognitiva*

La limitación en las capacidades sensoriales o en las habilidades para generar señales (como la voz y la escritura) disminuye o restringe la capacidad de interacción entre dos entidades que intenten comunicarse. Esto se ve claramente cuando una persona intenta hablarle a una persona con disminución auditiva. Probablemente no obtendrá una respuesta hasta que no capte su atención a través de otro canal sensorial, como la vista.

Cuando la interacción es cara a cara, es más fácil encontrar caminos alternativos para interactuar y comunicarse. Sin embargo, cuando la interacción es con el ordenador o a través del ordenador, entonces las cosas se complican.

El ordenador comúnmente utiliza pocos mecanismos para la transmisión de información, los cuales pueden saturarse con mucha facilidad. El principal es la interfaz gráfica y la secundaria es la interfaz auditiva. Aparte de estos dos medios existen otros menos usados como las interfaces táctiles y algunos entornos experimentales como AROMA [Pedersen and Sokoler, 1997] que utiliza el calor, la humedad y otros medios para transmitir información.

Los teléfonos móviles utilizan un mecanismo de interacción muy particular: el vibrador. Con este medio, se puede transmitir información a través del tacto variando la potencia y el ritmo de las vibraciones como si de música se tratara. La interfaz vibradora le da una ventaja al teléfono móvil sobre el ordenador, aunque el ordenador está diseñado para trabajar con una interfaz gráfica mucho mayor.

Cuando el ordenador resulta ser intermediario entre uno o varios individuos y su entorno, el ordenador debe capturar toda la información *de interés* del entorno y transmitirla a los individuos que requieren dicha información. Por lo tanto, la interacción entre los individuos y su entorno estará restringida a los medios de

percepción y transmisión de información soportados por el sistema de software que realiza la tarea de intermediario.

Los sistemas software se han convertido en los intermediarios más utilizados para prácticamente cualquier tarea. Cuando el entorno es hostil o peligroso, los ordenadores y otros dispositivos le permiten al ser humano interactuar con esos entornos. En la era de la información, la gran cantidad de aplicaciones software, sistemas de ventas o redes sociales han creado por sí mismas entornos virtuales los cuales generan la necesidad de interactuar con ellos a través del ordenador.

Los sistemas de software creados para el trabajo grupal o para la colaboración entre individuos deben soportar la interacción entre los usuarios y el sistema, y también la interacción y colaboración entre usuarios y los otros usuarios. Mucha interacción soportada a través de pocos canales.

Para reducir esta situación sin tener que utilizar más canales de comunicación se utilizan técnicas de filtrado de información dependientes del contexto y del usuario [Kirsch-Pinheiro et al., 2005, Ogata and Yano, 2000], así como técnicas específicas para la definición de la información que debe proporcionarse a un usuario [Drury and Williams, 2002].

Dado que ningún usuario es igual a otro, la combinación de las técnicas para una correcta definición de la información de Awareness requerida junto con técnicas de filtrado adaptadas o personalizadas para cada usuario puede ser la mejor alternativa si se dispone de recursos suficientes durante el desarrollo de la aplicación.

7.2.7. *Complejidad*

Un factor importante que crea un reto para la generación del Awareness en un usuario es el incremento de la complejidad en el sistema.

La complejidad en un sistema afecta negativamente al Awareness y a la carga de trabajo de un usuario [Sharit and Salvendy, 1982]. Esto se produce debido a

varios factores como el incremento de los componentes del sistema, el grado de interacción entre dichos componentes su dinánica o su tasa de cambios. Además, la complejidad de las tareas de un usuario puede crecer a través del incremento del número de objetivos o de subtareas y de la cantidad de decisiones que deben realizarse en relación con el sistema.

Esto también se refleja en las interfaces de usuario. Si una interfaz tiene una gran cantidad de componentes y cada uno tiene un comportamiento dinánico, la complejidad de la interfaz se verá incrementada y con ello sus efectos negativos sobre el Awareness del usuario.

Estos factores aumentan la carga de trabajo mental requerida para conseguir Awareness aunque sea en los primeros niveles. Cuando estos factores exceden la capacidad humana, la capacidad para obtener Awareness sufrirá las consecuencias.

7.3. Conclusiones sobre las dificultades en el soporte del Awareness

Hemos revisado varios aspectos del Awareness relacionados con el ser humano y con los sistemas software los cuales dificultan su soporte. Concluimos esta sección con algunas recomendaciones sobre como tratar cada uno de estos aspectos, ya que de una forma o de otra están presentes en cada uno de los entornos en donde se utiliza el Awareness para la toma de decisiones. En el caso que nos ocupa, los sistemas Groupware, estas recomendaciones pueden servir durante el diseño del sistema, durante su mantenimiento y durante el entrenamiento y capacitación de sus usuarios, quienes al final pondrán sus necesidades sobre la mesa en lo que respecta al Awareness que el sistema debe proporcionarles.

En lo que respecta a la preatención, sabiendo que permite recibir información sin requerir demasiada atención, podemos afirmar que el uso de mecanismos para fomentar la pre-atención puede aliviar en gran medida la sobrecarga cognitiva relacionada con la saturación de algunos canales de transmisión de información, como es el caso del visual.

Para fomentar la atención a determinados elementos de Awareness, se requiere primeramente un entorno libre de agentes estresantes. En ese entorno se puede fomentar la atención usando señales más "*fuertes* " para los datos más importantes, además de que se pueden reducir o suavizar los otros elementos de Awareness que se proporcionan paralelamente.

El entrenamiento y la experiencia son fundamentales para una buena percepción. Sin embargo, no todos pueden permitirse un entrenamiento profundo y una experiencia inmediata. Una introducción progresiva sería lo más adecuado, pero en muchas ocaciones no es posible porque implicaría un sistema muy grande. Dependiendo de las necesidades de los usuarios y de los recursos disponibles se puede encontrar el equilibrio entre un desarrollo extenso y un entrenamiento profundo.

Los usuarios deben tener presentes cuales son sus objetivos en cada tarea. Por parte de los desarrolladores, estos objetivos deben apoyarse a través de los mecanismos de Awareness que se implementen en el sistema. Es decir, el Awareness estará en función de los objetivos de los usuarios en cada tarea.

Reducir los factores que provocan estrés es un aspecto importante en el diseño de sistemas que requieren Awareness. Los internos al ser humano se tratan con entrenamiento, buenos hábitos y práticas empresariales que estimulen la calma y serenidad en los usuarios. El estrés referente al sistema puede tratarse reduciendo la complejidad de las tareas, la cantidad de información transmitida a los usuarios o canalizándola a través de vías de comunicación alternativas.

La complejidad puede reducirse con entrenamiento y con un buen diseño del sistema, el cual ayude a los usuarios a centrarse en los elementos importantes sin aumentar demasiado el trabajo mental y apoyando los estados superiores del Awareness (comprensión y proyección). Las interfaces de usuario y el flujo de interacción y ejecución de tareas juega un papel muy importante es este aspecto, ya que la complejidad puede atacarse simplificando y dividiendo las tareas complejas.

8. Resumen sobre el estado general el Awareness

A fin de proporcionar una visión general del Awareness y sus aplicaciones presentamos un resumen de una parte de la investigación realizada en este capítulo.

Un catálogo de varios tipos de Awareness analizados en la sección 4 se muestra en la Tabla 9, destacando los autores con trabajos importantes en la investigación de dicho tipo de Awareness y una pequeña descripción de cada uno de ellos.

Un tipo de Awareness proporciona información de alguna o algunas entidades del entorno. No obstante, la información que proporciona puede variar dependiendo del dominio de trabajo, de las entidades observadas y de la información manipulada.

Por ejemplo, el Awareness de Localización proporciona la información de la localización de una o varias entidades de interés. Sin embargo, la localización puede representarse de varias maneras y transmitirse al usuario de acuerdo a sus necesidades. No es lo mismo la localización de un coche a la localización del puntero en el ordenador o a la localización de un avión en vuelo. Aunque se trata del mismo tipo de Awareness, la representación explícita de cada "*elemento de información*" o de Awareness es dependiente del objeto o concepto que se *observa*.

En la Tabla 10 presentamos una descripción general de los elementos de información que componen cada tipo de Awareness de acuerdo a su uso general.

Cualquier sistema que soporte el Awareness se encontrará con sus dificultades inherentes. En la Tabla 11 mostramos un resumen de las mismas así como varias técnicas para reducirlas.

Varias de las dificultades del soporte del Awareness están relacionadas con el sistema que proporciona Awareness y con su comportamiento. Si los diseñadores toman en cuenta el Awareness desde el principio pueden anticiparse a los efectos colaterales derivados del soporte del Awareness, como es el aumento de la complejidad en el comportamiento del sistema y en la interacción con los

Tipos de Awareness	
Awareness	**Descripción**
De situación	Se utiliza para la toma de decisiones en entornos dinámicos y complejos: Entornos militares, gestión de emergencias, monitorización y colaboración en entornos altamente dinámicos [Endsley, 1995, Endsley et al., 2003, Drury et al., 2006].
De espacio de trabajo	Es el Awareness necesario para poder trabajar en un espacio de trabajo compartido con otros usuarios. Incluye varios tipos de Awareness más sencillos como el Awareness de presencia, el Awareness de localización, el Awareness de Grupo, etc. [Gutwin and Greenberg, 2002].
De grupo	Importante para conocer la estructura del grupo de trabajo durante una tarea grupal [Sohlenkamp, 1999, Gutwin et al., 1995].
De actividad	Necesario para soportar la ejecución de actividades grupales. También se refiere al Awareness de las actividades que realizan los otros miembros del equipo [Carroll et al., 2003, Carroll et al., 2006].
Del contexto	Es el Awareness que se proporciona en entornos en donde el contexto del usuario o del sistema es de interés, como en los sistemas para dispositivos móviles o en la robótica [Kirsch-Pinheiro et al., 2005, Baldauf et al., 2007, Liechti, 2000].
Periférico	Es el Awareness que se proporciona usando interfaces de usuario periféricas o técnicas que requieren poca concentración o enfoque para trasmitir información [Liechti, 2000, Matthews et al., 2007].
Del conocimiento	En entornos de E-learning[18] se utiliza para saber cómo se maneja el conocimiento, quién busca conocimiento y quién crea conocimiento [Ogata et al., 1996].
Del conocimiento compartido	Es el Awareness del conocimiento compartido y de la construcción del conocmiento compartido. También está relacionado con los sistemas de aprendizaje electrónico [Collazos et al., 2003].
De la presencia	Permite conocer si un determinado ente (usuario, artefacto, agente, etc.) se encuentra presente en el sistema, espacio de trabajo, contexto compartido o algún otro contenedor de interés [Gutwin and Greenberg, 2002, Carroll et al., 2003].
De la disponibilidad	Permite conocer la disponibilidad de un ente para realizar una actividad o tarea. Por ejemplo, un usuario puede estar *presente* en la sala de *chat* pero no estar disponible para tener una conversación [Gutwin and Greenberg, 2002].
Del horario	Referente a la disponibilidad futura de una o varias entidades (usuarios por ejemplo) para realizar una determinada tarea en el futuro. Por ejemplo, un calendario de días y horas en las que un grupo de usuarios expresan sus horas disponibles para tener reuniones del grupo [Kim and Kim, 2007].
Del ritmo	Se refiere al Awareness sobre los momentos de disponibilidad que tendrá un ente en el futuro cercano. La diferencia con el Awareness de horario es que la propuesta de [Begole and Tang, 2007] se basa en los tiempos pasados de disponibilidad de los usuarios para realizar alguna tarea.
Del cambio en documentos	Proporciona el conocimiento sobre los cambios ocurridos en un documento compartido y escrito por diferentes usuarios [Tam and Greenberg, 2006].
Anticipado	Se refiere a las expectativas que se pueden tener sobre las acciones de otros usuarios [Prinz et al., 2010].
De la emoción	Se refiere al conocimiento del estado emocional de otros usuarios o miembros del equipo de trabajo [Neyem et al., 2007].
De la dificultad	El trabajo de [Carter and Dewan, 2010] se refiere al Awareness de la dificultad de realizar una tarea de programación. Sin embargo, este concepto puede usarse para representar el conocimiento de la dificultad de realizar alguna tarea determinada.

Tabla 9. Tipos de Awareness relacionados con los sistemas Groupware.

Información relevante de cada tipo de Awareness	
Awareness	**Información**
De situación	Elementos relevantes de la situación y del contexto, los cuales afectan el proceso de toma de decisiones durante la ejecución de tareas.
De espacio de trabajo	Información del grupo de trabajo y su estructura, información de cada usuario como la presencia, disponibilidad, actividad actual, localización en el espacio de trabajo compartido, rango de visión, alcance de sus movimientos, entre otros.
De grupo	Información de los usuarios o entidades que componen el grupo, así como datos detallados de cada uno de ellos.
De actividad	Información referente a la actividad que se realiza de forma individual o en grupo: duración, objetivos que se están cumpliendo o que se cumplirán al terminar la tarea, su dificultad, restricciones, colaboradores actuales, los colaboradores que se desempeñan mejor en dicha tarea, entre otra información que apoye la efectividad de los usuarios para llevar a cabo dicha tarea.
Del contexto	Dependiendo de la tarea a realizar, el contexto puede ser el del usuario o usuarios que desempeñan la tarea, o puede ser el contexto del dispositivo que proporciona Awareness, o del propio sistema. El contexto puede ser conceptual (ideas que definen otra idea), físico (aspectos físicos como la localización, la temperatura, la posición relativa, etc.) o social (acompañantes, amigos, grupo de trabajo).
Del conocimiento	Información referente a quíen accede/busca/agrega conocimiento en el sistema.
Del conocimiento compartido	Información sobre como cada usuario utiliza y agrega conocimiento al sistema. También es la información sobre qué usuario conoce más de un determinado tema, quién le ayudó a desarrollar ese conocimiento, de dónde proviene un determinado conocimiento, etc.
De la presencia	Uno o varios identificadores y datos de los usuarios o agentes y si se encuentran presentes o no en el sistema.
De la disponibilidad	El nivel de disponibilidad de un sistema. Puede ser un *disponible* o un *no disponible*, o estados intermedios como *disponible por X tiempo*, *disponible solo si es urgente*, etc.
Del horario	El identificador del usuario o agente y el conjunto de periodos temporales en los cuales puede realizar una determinada tarea (una reunión por ejemplo).
Del ritmo	Información sobre el usuario o los usuarios de interés junto con sus posibles espacios temporales de presencia o disponibilidad para realizar alguna tarea individual o grupal.
Del cambio en documentos	Información del autor del cambio, datos temporales y una descripción del cambio realizado.
Anticipado	Información de identificación de los usuarios de interés junto con información de la tarea que posiblemente realize en un periodo de tiempo determinado, casi como un pronóstico de actividades.
De la emoción	Información de los usuarios de interés junto con su estado emocional, como puede ser tristeza, felicidad, risa, enojo, etc.
De la dificultad	Identificación de los usuarios de interes, de la tarea en proceso y de la dificultad que se tiene en dicha tarea.

Tabla 10. Información relevante que proporciona un tipo de Awareness.

Dificultades en el soporte del Awareness	
Dificultad	**Técnicas de control**
Atención	Para atraer la atención se utilizan señales que denoten cambio y que sean llamativas al sentido correcto del sistema. Si las interfaces de usuario producen aburrimiento entonces decrementan la utilidad del Awareness.
Percepción	Se refiere a la percepción de información, la está afectada por razones individuales como el entrenamiento, los objetivos del sistema, los objetivos propios, etc.
Memoria de trabajo	Un aspecto humano que es afectado si se sobrecarga de forma sistematizada. La memoria de trabajo es donde permanece la información de Awareness y en donde se sube a los distintos nivel de conciencia.
Memoria de largo plazo	Incluye el conocimiento del sistema, sus particularidades y la experiencia propia sobre uno mismo. El entrenamiento y un buen diseño de las interfaces de usuario son necesarios para los entornos con sistemas muy cambiantes.
Objetivos	Si no se tienen claros los objetivos en el sistema y la parte en la que el Awareness los apyoa, entonces tal vez no puedan servir a su cometido.
Privacidad	La privacidad en el aspecto del Awareness se refiere a no poder acceder a información que no se puede leer por el usuario y que no necesita. Se necesitan politicas de acceso a datos que sean flexibles para encontrar el equilibrio entre lo que se puede acceder y lo que no se puede acceder.
Sobrecarga	Se refiere a la sobrecarga mental ocasionada por recibir grandes cantidades de datos. Para reducir este problema existen herramientas de filtrado, políticas de acceso a datos y el uso de sentidos alternos a los más saturados actualmente: la visión y la audición.

Tabla 11. Dificultades inherentes al soporte del Awareness en un sistema de software.

usuarios.

En las metodologías de desarrollo de sistemas Groupware revisadas en el Capítulo 2 no se contemplan dichas dificultades. Siendo que dichas metodologías siguen un enfoque dirigido por modelos y que hasta ahora no existe un modelo del Awareness apropiado para ser utilizado en el MDD, esta situación es comprensible. Sin embargo, la necesidad de dichos modelos se hace latente al reconocer que las dificultades del soporte del Awareness no son tratadas como parte del proceso de desarrollo de los sistemas Groupware y del software en general.

9. Conclusiones

El Awareness como estado mental es un componente necesario para la interacción de un individuo con su entorno y/o con otros individuos, ya que le permite tomar decisiones coherentes con el entorno en el que se desenvuelve.

Podemos afirmar que la interacción sin Awareness se convierte en un conjunto de acciones desligadas que pueden ser tan efectivas como desastrosas, ya que no son planificadas para ejecutarse en un entorno conocido al momento.

Sin Awareness no hay interacción efectiva. Sin interacción efectiva difícilmente puede llevarse a cabo la colaboración o la cooperación entre distintos usuarios o agentes.

Cada tipo de entorno, situación o entidad puede proporcionar información específica que permite formar un tipo de Awareness u otro. De ahí la cantidad tan grande de definiciones del Awareness, las cuales repercuten en la dispersión e incompatibilidad conceptual de las varias técnicas y herramientas para el soporte del Awareness. La definición más genŕica y aplicable conceptualmente a cualquier tipo de Awareness la encontramos en el trabajo de Endsley [Endsley, 1995], en el cual se define el Awareness de Situación.

Una base conceptual y teórica como la propuesta por Mica Endsley no ha sido utilizada previamente para modelar el Awareness en general. De hecho, dicha de-

finición es muy poco comentada en la lituratura consultada sobre el Groupware, lo cual muestra que dicha comunidad se ha orientado hacia el soporte del Awareness del Espacio de Trabajo, el cual es necesario en los procesos de colaboración pero no es el único que puede requerirse en un proceso colaborativo/cooperativo.

Existen varios trabajos que describen el soporte del Awareness en distintas etapas del desarrollo de un software, aunque muy pocos a nivel de requerimientos o a nivel de modelado. Además, muy pocos buscan mantener una trazabilidad entre las distintas etapas en las que interviene el Awareness. Esto puede deberse a que no está claro como soportar el Awareness y en cuales etapas del desarrollo. El problema de la trazabilidad es solo una consecuencia de esta situación.

Otra problema detectado es la especialización de las técnicas de soporte del Awareness, ya que varias se enfocan a unos pocos tipos de Awareness, dejando fuera otros tipos que pueden ser importantes para el dominio del sistema.

Los sistemas que solo soportan el concepto de *feedback* o retroalimentación olvidan que el proceso de retroalimentación se debe en parte a la necesidad de Awareness que tienen los usuarios con respecto a las tareas que desarrollan en el sistema. Podemos afirmar que la necesidad de Awareness justifica la utilización de mecanismos de *feedback*.

En algunas metodologías específicas para el Groupware se soporta el Awareness implícita y explícitamente. Sin embargo, encontramos algunos puntos que pueden mejorarse:

- No se especifican formas para definir el Awareness como información. Es decir, no se explica como definir el Awareness en el sistema de forma que pueda ser útil. Por lo tanto, se requiere un modelo explícito para definir con exactitud qué información va a ser transmitida a los usuarios. Dicha información representa el Awareness que van a obtener a través de la percepción constante.
- El modelo del Awareness debe ser genérico en el sentido que permita definir

cualquier tipo de Awareness. La idea en ésto es que se debe poder definir cualquier Awareness que se requiera.

- El Awareness debe manejarse como parte del proceso de desarrollo. Esto se debe a que en mayor o menor medida, todos los sistemas actuales proporcionan algún tipo de Awareness.
- Reafirmar la posición del Awareness como requerimiento para la ejecución de algunas tareas del sistema.
- El Awareness puede estar ligado a una tarea de cualquier tipo (individual, colaborativa o cooperativa), además de a un conjunto de usuarios en una determinada situación. Es decir, un requerimiento básico de Awareness puede expresarse como "*el usuario/rol/grupo require el Awareness X para realizar la tarea T si se cumple una determinada condición*".
- Si se implementan mecanismos de Awareness y se generan o diseñan interfaces de usuario para su soporte, es necesario mantener la trazabilidad desde dichas interfaces hasta las necesidades de Awareness que las produjeron.
- Proporcionar la flexibilidad para adminitir procesos de filtrado o personalización durante la selección/transmisión de información de Awareness debido a la sobrecarga cognitiva que puede causar o a la irrelevancia de los datos enviados a los usuarios.

Existen muchas interfaces para proporcionar Awareness. Lo que las define es que deben transmitir datos de forma periódica (en general) y que están ligadas a un proceso de toma de decisiones. Se distinguen principalmente las interfaces para monitorizar datos, las interfaces informativas/interactivas y las interfaces Groupware, las cuales añaden la dimensión que representa a los demás usuarios del grupo o grupos de trabajo, ya sean estáticos o dinámicos.

Varias técnicas generan interfaces de usuario a partir del modelo de tareas o de otros modelos de actividades. Ciertamente hacen falta catálogos más amplios para abarcar interfaces de usuario orientadas al Awareness que cumplir los

requisitos específicos para cada uso. Ésto no es una tarea fácil y es que hay entornos como la Web por ejemplo o los mundos virtuales en donde prácticamente cualquier cosa puede transmitir información de Awareness.

No puede negarse que crear un componente de UI es complicado y más si tiene un comportamiento complejo y fuertemente ligado con el sistema. A pesar de ello es necesario tener presente la justificación y el objetivo de cada uno de estos componentes y poder llegar hasta las acciones o requerimientos que representan. De esta forma el mantenimiento o cambio de estos componentes de la UI será más manejable y por consiguiente, mejor para los desarrolladores y para los clientes.

La generación del Awareness está condicionada por varios aspectos internos y externos al ser humano, así como internos y externos al sistema que lo desea soportar. Cualquier sistema que pretenda soportar el Awareness deberá contemplar las dificultades que acarrea su integración, las cuales pueden mitigarse con entrenamiento y un buen diseño del sistema, así como mecanismos de Awareness flexibles y adaptables a las individualidades presentadas por cada usuario.

Estas conclusiones nos permiten centrarnos en las necesidades que detectamos como primordiales para mejorar el soporte del Awareness en el desarrollo de sotfware, especialmente en el desarrollo de sistemas Groupware:

- Un modelo para definir cualquier tipo de Awareness de acuerdo a alguna teoría bien fundamentada sobre el Awareness, su significado y sus componentes.
- Una guía o marco de referencia para guiar el soporte del Awareness y para integrar el uso del modelo anterior en la metodología de trabajo o de desarrollo.
- Una forma específica para representar un requerimiento de Awareness acorde con el marco de referencia o guía propuesto anteriormente.

Dichas aportaciones se presentan en el siguiente Capítulo.

CAPITULO 4

Modelado del Awareness

1. Introduction

Como hemos visto en capítulos anteriores, el Awareness está integrando de forma implícita en el desarrollo de software a pesar de no haber un consenso ampliamente aceptado de lo que es el Awareness y de como soportarlo.

Las técnicas para soportar el Awareness (y cualquier otro requerimiento) utilizan una base teórica y conceptual para guiar su integración en el desarrollo del software. En el caso del Awareness, el haber una gran cantidad de definiciones (Capítulo 3 Sección 2) ha generado un conjunto de técnicas para el soporte del Awareness diferentes desde la propia base conceptual, dificultando su integración en conjunto con las metodologías de desarrollo.

En este trabajo se ha considerado que tener un soporte completo del Awareness implica manejar tres aspectos:

- La descripción del requerimiento de Awareness como necesidad de información.
- La descripción del propio Awareness como información.
- Las relaciones entre los distintos componentes del sistema que soportan dicha necesidad de información.

En el Capítulo 3 Sección 5.2 hablamos sobre el Awareness como requerimiento de información, lo cual consideramos que debe integrarse en las metodologías de desarrollo, ya que es un requerimiento crucial en la modernidad de los sistemas sociales y de gran escala.

En el Capítulo 3 Sección 5.4 presentamos algunos avances en la integración del Awareness en algunas metodologías de desarrollo específicas para sistemas colaborativos, en donde la necesidad de soportar el Awareness es más evidente. Sin embargo, como pudimos comprobar, dicho soporte no es completo, siendo específico a ciertos tipos de Awareness y a ciertas técnicas de interacción, con lo cual al final tenemos un soporte limitado y poco flexible.

Encontrándonos con este panorama, hemos desarrollado un marco de referencia para el soporte del Awareness, un modelo conceptual del Awareness, un método para su caracterización y una guía para representar sus requerimientos.

Estas aportaciones se apoyan en cada uno de los trabajos revisados, ya que de manera individual todos ellos aportan una visión importante de lo que es el Awareness y por qué es importante soportarlo de manera efectiva. Como dice la frase, "nos apoyamos sobre hombros de gigantes", o en algunos casos, sobre varias aportaciones apiladas para ir más alto.

En las siguientes secciones describiremos los modelos de los tres componentes previamente identificados. Primeramente el marco de referencia sobre el soporte del Awareness (Sección 2), el cual proporcionará las bases para integrar el Awareness al ciclo de desarrollo de un software. Después, el modelo old: de caracterización del Awareness (Sección 3), el cual permite representar la estructura de la información de Awareness, así como su integración con las fuentes de datos del propio sistema. Para finalizar describiremos nuestra formalización del Requerimiento de Awareness (Sección 4) basada en el marco de referencia del soporte del Awareness, descrito anteriormente.

Posteriormente a estas secciones presentaremos el ejemplo general de aplica-

ción, el cual mostrará la utilización de los modelos propuestos en un entorno de investigación actual.

2. Marco de referencia para el Soporte del Awareness

El soporte del Awareness se refiere a las técnicas y elementos del desarrollo de software que permiten describir, modelar, manipular y ejecutar los mecanismos para proveer Awareness en un sistema.

Desde una perspectiva de ingeniería del software, cualquier característica implementada en un software proviene de uno o varios requerimientos, ya sean funcionales o no funcionales. Normalmente la definición de estos requerimientos comienza en las fases tempranas del desarrollo y continua a lo largo del desarrollo. Una vez identificados estos requerimientos se comienza a trabajar con ellos, ya sea en documentación, modelado, diseño o directamente en la codificación. La creación de nuevos prototipos o la retroalimentación de los usuarios puede generar nuevos requerimientos y nuevas fases de modelado, diseño e implementación de los mecanismos para cumplir dichos requisitos.

Los trabajos analizados en esta área (Capítulo 3 Sección 5) nos han llevado a orientar nuestra aportación hacia el soporte conceptual del Awareness, el cual consideramos que es una necesidad importante debido a la considerable cantidad de herramientas y técnicas existentes para el soporte del Awareness, las cuales tienen una teoría muy específica de fondo, pero comparten muchas similitudes en cuanto a su propósito, estructura y funcionamiento.

Uno de los objetivos de este trabajo es la definición de un marco de referencia que permita tomar una metodología de desarrollo y comprobar su soporte y manejo del Awareness. A simismo, el marco de referencia debe guiar la integración del Awareness como requerimiento de información y su soporte a lo largo de toda la metodología, como mostraremos el Capítulo 5.

En la siguiente sección describiremos el marco de referencia para el soporte

5Ws y una H sobre un concepto o evento.	
Pregunta	**Descripción**
¿Quién? (*Who?*)	De quién se habla o quién está involucrado.
¿Qué? (*What?*)	De qué se habla o qué es lo que pasó.
¿Cuándo? (*When?*)	Cuándo toma lugar el evento.
¿Dónde? (*Where?*)	Dónde toma lugar el evento.
¿Porqué? (*Why?*)	Por qué sucede el evento.
¿Cómo? (*How?*)	Cómo sucede el evento.

Tabla 12. Descripción de las 5 Ws y una H sobre un evento o concepto.

del Awareness, el cual utilizamos para proporcionar una guía en la integración del soporte del Awareness, así como tambien en el desarrollo de los modelos del Awareness propuestos en las secciones 3 y 4 de este capítulo.

2.1. Marco de referencia

Otros autores como Carl Gutwin han utilizado las 5 W's y 1 H[1] (Tabla 12) para describir sus marcos de referencia parciales para el soporte de algún tipo de Awareness, como es el caso del Workspace Awareness [Gutwin and Greenberg, 2002].

Para describir nuestro marco de referencia utilizaremos la misma metodología usando las 5 Ws y una H, ya que resulta familiar y fácil de entender.

Las seis preguntas van dirigidas a describir la necesidad o requerimiento de Awareness en un sistema informático y su repercusión en el desarrollo del sistema, ya que para soportar dichos requerimientos deberán diseñarse e implementarse mecanismos y procesos en el sistema, culminando en interfaces de usuario.

Un requerimiento de Awareness es la representación de la necesidad de información obtenida de un entorno particular por parte de un grupo delimitado de usuarios, los cuales requieren dicha información para la ejecución de tareas individuales o grupales, y que es soportado por un sistema software ya que provee al usuario o agente la información actualizada y constante requerida para soportar

[1] *Who/Quién, What/Qué, Where/Dónde, When/Cuándo, Why/Porqué, How/Cómo*, `http://es.wikipedia.org/wiki/Cinco_W`

dicho Awareness.

Las 5Ws aplicadas un requerimiento de Awareness como una necesidad especial de información se muestra a continuación:

- *¿Quién?*
  - Los usuarios o actores del sistema que requiren la información de Awareness para su proceso de toma de decisiones durante la realización de una tarea particular. Puede ser un grupo estático o dinámico, un rol, un individuo, etc. Eso dependerá de la estructura del propio sistema.

    Un usuario puede en cierto momento tener un requerimiento de Awareness, pero dada su alta sobrecarga cognitiva en ese momento puede no ser apto para recibir dichos datos de Awareness. La sobrecarga cognitiva se trató en el Capítulo 3 Sección 7.2.6, y puede restringir la aplicación de un requerimiento, ya que todos los usuarios reaccionan y asimilan de forma distinta a los mismos estímulos y es necesario evitar la sobrecarga para no afectar su interacción con el sistema y con los otros usuarios.
- *¿Qué?*
  - La información de Awareness que será proporcionada a los actores para cumplir el requerimiento. La información de Awareness será identificada, modelada y extraída de las entidades del dominio y/o del sistema (hardware, variables del entorno o contexto, etc.), aunque en diferentes situaciones podemos observar como son los propios requerimientos de Awareness los que generarán entidades derivadas para organizar dicha información, como es el caso de requerimientos de información compleja.

    Para definir la información de Awareness proponemos en esta tesis un modelo específico que permite modelar de forma general cualquier tipo de Awareness en base a las entidades del dominio y del sistema, con

lo cual se integra la información de Awareness al conjunto de modelos para describir un sistema software.

- Las restricciones de acceso a los datos que conforman la información de Awareness. Por una parte, puede manejarse como política de privacidad, por otra, como política de control de sobrecarga cognitiva. Sin embargo, las restricciones de acceso están más relacionadas con la privacidad que con la sobrecarga cognitiva, lo que implica otras restricciones orientadas a individuos particulares o grupos semejantes.

- *¿Porqué?*

  - Los requerimientos de información, los cuales pueden incluir una descripción informal. Es importante que los requerimientos expresen las 5 W's (*who, what, whay, where, when*) para obtener toda la información del requerimiento. Para el soporte del Awareness, la respuesta a la pregunta *¿Porqué?* justifica el soporte de todos los mecanismos para cumplir dicho requerimiento. Además proporciona documentación importante a nivel de desarrollo y a nivel de usuario.

- *¿Dónde?*

  - La tarea en donde el Awareness es requerido por los actores para interactuar correctamente con el sistema o con otros actores. Por ejemplo, durante la edición de un documento, es necesario saber si el documento tiene cambios sin guardar. La tarea en donde se requiere este tipo de Awareness es la edición del documento, no el inicio de sesión ni cualquier otra tarea.

    Si la tarea en donde se requiere Awareness tiene varias subtareas, entonces se deberá proporcionar dicho Awareness durante la realización de las subtareas, a menos que las restricciones de aplicación dicten otra cosa.

- *¿Cuándo?*

- El estado del sistema, dominio o contexto en el cual el Awareness es requerido. Es decir, las restricciones de aplicación del requerimiento basadas en el estado del sistem/dominio/contexto en ese momento. Por ejemplo, un tipo de Awareness puede ser requerido en una tarea sólo si el hardware del cliente soporta una determinada tecnología.

  Las restricciones de aplicación pueden llegar a ser muy complejas y pueden llegar a estar ligadas a muchas situaciones y estados del sistema o hasta a otras restricciones de otros requerimientos de Awareness. Puede ser necesario utilizar técnicas especiales para manejar estas restricciones si es que se pretenden utilizar en el MDD. Una ventaja de representarlas o ligarlas a los requerimientos de Awarenes es que pueden trazarse desde su origen hasta todos los mecanismos de soporte que utilicen estas restricciones.

- *¿Cómo?* Esta pregunta puede hacerse principalmente de dos maneras: ¿Cómo se requiere la información de Awareness? y ¿Cómo se soporta el requerimento de Awareness?. La primera pregunta está orientada a generar una descripción de las técnicas o mecanismos de interacción preferidas o requeridas por los usuarios. Por ejemplo, un determinado requerimiento de Awareness puede estar orientado a las personas ciegas, las cuales requieren un proceso de interacción distinto y un mayor o total énfacis en las interfaces de usuario táctiles y auditivas. Este tipo de preferencias pueden plasmarse también en el modelado del usuario y formar parte del motor de adaptación y personalización [Ogata and Yano, 2000].

  La descripción de las preferencias de interacción no tiene por qué ser informal. Pueden utilizarse lenguajes formales o descriptivos para que dichas especificaciones sean usadas durante el proceso de desarrollo. De nuevo la capacidad de definir estas preferencias o restricciones desde el principio puede permitir automatizar muchas partes del desarrollo, siempre y cuando se cuente con la tecnología adecuada para ello.

La segunda pregunta está orientada a las relaciones entre los distintos modelos y componentes que participan en el soporte del requerimiento en cuestión. Algunos de estos elementos son los siguientes:

- La tarea, proceso o procedimiento encargados de proporcionar la información de Awareness a través de las interfaces de usuario, ya sea como primer objetivo o como objetivo secundario.
- Las relaciones entre los elementos del modelado (o del código) directamente relacionados con el soporte del Awareness. Estas relaciones forman la estructura del soporte del Awareness como una red que permite trazar los requerimientos de Awareness de principio a fin. Dicha estructura puede ser utilizada por herramientas externas o de forma manual para crear pruebas, validar aspectos varios sobre el soporte e implementación, documentar para desarrolladores y usuarios externos, etc.

La Figura 41 muestra varios elementos comunes en el modelado de sistemas de software incluyendo también las entidades que representa el Awareness, como son los elementos concretos y los tipos de Awareness. El elemento central es el requerimiento de Awareness el cual se relaciona de forma particular con los distintos elementos del modelado, de tal forma que puede trazarse un camino desde los propios requerimientos de Awareness hasta todos los modelos que lo soportan de alguna manera.

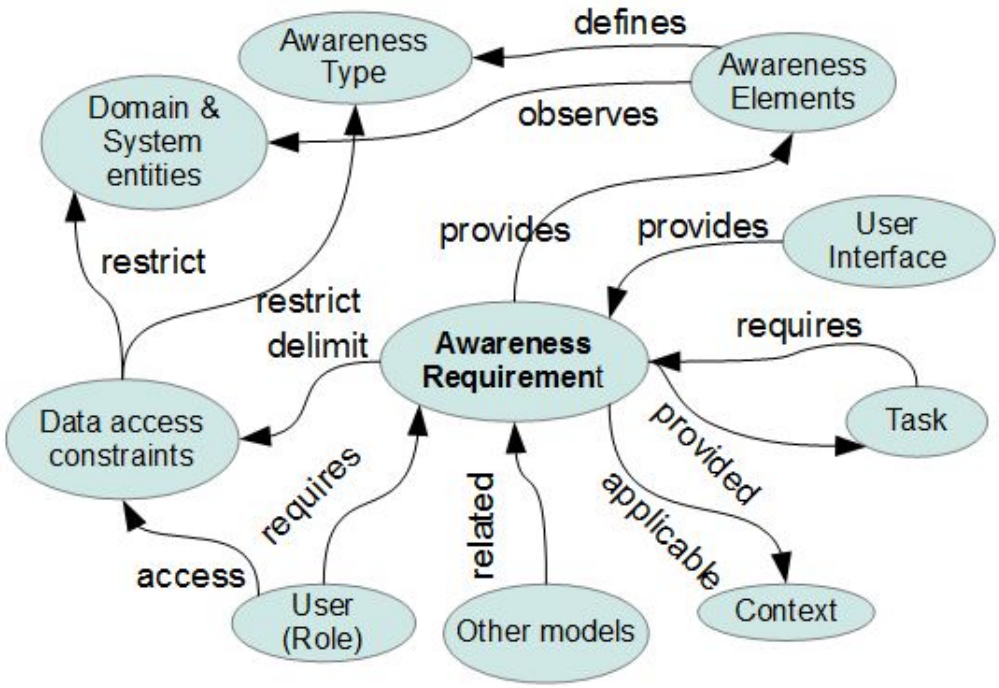


Figura 41. Mapa conceptual de las entidades que intervienen en el soporte del Awareness en un sistema software.

3. Modelo del Awareness y método de caracterización

Una de las cuestiones más importantes en lo que respecta al soporte del Awareness es la representación y caracterización del Awareness como información requerida por los usuarios o actores del sistema. Dada la cantidad y flexibilidad de los tipos de Awareness analizados (Capítulo 3 Sección 4) concluimos que los tipos de Awareness son representaciones genéricas de la información requerida por los usuarios, pero que además son dependientes del dominio en donde es requerida dicha información. Por tanto, cada tipo de Awareness puede variar entre dominios de aplicación aunque su objetivo y su estructura general permanezcan iguales.

Si en cada dominio el tipo de Awareness específico cambia pero la estructura general permanece, creemos que lo más apropiado es proporcionar los modelos necesarios para representar los tipos de Awareness y un método sencillo y flexible para caracterizarlos en el dominio de trabajo o sistema específico.

Caracterizar un tipo de Awareness en un dominio significa darle una estructura específica ligada a dicho dominio, de tal forma que el tipo de Awareness represente una estructura de información (modelo del Awareness) y el origen de los propios datos (enlaces con el dominio). Es decir, darle una funcionalidad similar a la que tiene una Tabla en el modelo Entidad-Relación o una Clase en el modelo orientado a objetos.

La caracterización del Awareness es una forma de instanciar elementos del Awareness en base a la definición previa de los tipos de Awareness (sus características y su estructura).

Para modelar los tipos de Awareness proponemos un modelo conceptual genérico del Awareness orientado al desarrollo de software. . Sin embargo, el modelo del Awareness está expresado usando como base los términos de la teoría del Awareness utilizada como base para este trabajo.

El modelo conceptual del Awareness está basado en la Teoría del Awareness de Situación (SA) [Endsley, 1995], la cual define el Awareness de Situación como:

"La percepción de los elementos del entorno en un volumen de espacio y tiempo, la comprensión de su significado y la proyección de su estado en un futuro cercano" [Endsley, 1995].

Analizando cada uno de los componentes del Awareness según la definición del Awareness proporcionada por Endsley podemos corroborar su viabilidad para usarse como teoría base para el desarrollo de los modelos conceptuales del Awareness.

En primer lugar, tenemos el nombre de éste Awareness en particular: Awareness de Situación. La cuestión clave es ¿qué situación? En pocas palabras se refiere a la situación actual.

Cualquier proceso de interacción o colaboración, bien sea con un entorno complejo, con un grupo de trabajo o con ambos, encierra una gran cantidad de decisiones que deben tomarse durante el curso de dicha interacción. Estas

decisiones dependen en gran medida del Awareness de situación en general.

Durante el proceso de toma de decisiones, la situación de interés representa el contexto que afecta positiva o negativamente a la propia toma de decisiones. Es decir, la información actualizada de lo que está pasando y que repercute en las decisiones tomadas.

El Awareness de Situación representa el estado de conocimiento generado por percibir, comprender (y tal vez proyectar) los valores de las características o atributos del entorno. Estos atributos del entorno son de utilidad al observador, el cual puede ser un ente biológico o un agente artificial.

La interacción entre agentes humanos y artificiales puede verse claramente en las técnicas de iniciativa mixta [Tecuci et al., 2007], las cuales forman parte de los sistemas de apoyo a la toma de decisiones usados en diversas áreas desde el manejo empresarial hasta la gestión de fronteras y migración [Castillo et al., 2010].

En las siguientes secciones describimos el modelo del Awareness y sus componentes según la teoría del Awareness de Situación.

3.1. Modelos del Awareness

El modelado conceptual del Awareness permite describir los elementos y las relaciones que conforman la estructura del Awareness.

Los modelos del Awareness que a continuación presentamos están orientados al proceso de desarrollo de software, y particularmente al desarrollo diridigo por modelos. No obstante, los modelos presentados pueden aplicarse al desarrollo de software tradicional. Sin embargo, creemos que el desarrollo dirigido por modelos sustituirá con el tiempo al desarrollo de software tradicional, especialmente en el desarrollo de sistemas complejos como son los sistemas Groupware.

La utilización de los modelos conceptuales del Awareness se explica con más detalle en el ejemplo de la sección 5 y en el Capítulo 5, en donde se detalla la integración de los modelos del Awareness en UsiXML [Limbourg et al., 2005], un

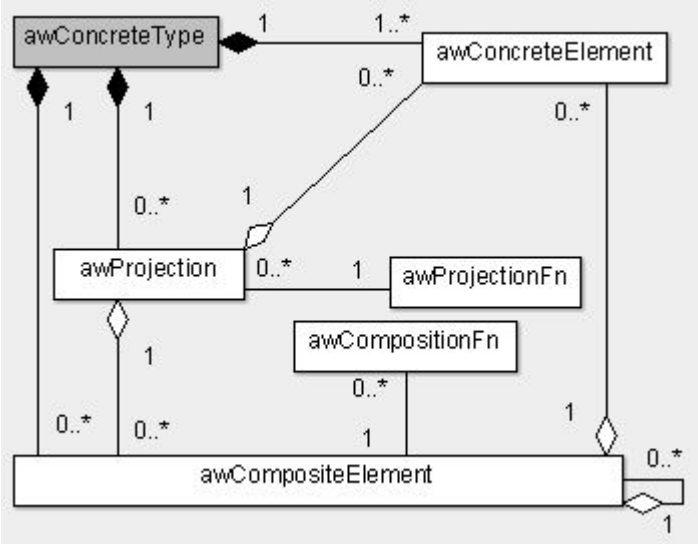


Figura 42. Modelo conceptual del Awareness en el que se incluyen los tres componentes descritos por [Endsley, 1995]: Percepción de los elementos, comprensión y proyección.

popular lenguaje de descripción de interfaces de usuario (UIDL) el cual permite desarrollar un sistema interactivo usando un enfoque dirigido por modelos.

Como elemento principal del modelo del Awareness tenemos el tipo de Awareness (*awConcreteType*, Figura 42), el cual representa el estado de conocimiento resultante de recibir constantemente información actual de ese tipo de Awareness. Es decir, un tipo de Awareness permite formar un estado de conocimiento, el cual está determinado por los elementos que componen el tipo de Awareness; por ejemplo, podemos tener Awareness de la Localización del Usuario, Awareness de la Localización de nuestro destino durante un viaje, etc.

Un tipo de Awareness está constituido por un conjunto de elementos concretos (*awConcreteElement*, Figura 43) que representan atributos o características de alguna entidad o entidades del entorno; por ejemplo, en el caso del *Location Awareness*, el elemento principal es *Location*, y un segundo elemento sería el objeto *Target* de esa localización. Estos elementos se distinguen por ser obtenidos de la entidad de la que proceden a través de los mecanismos sensoriales del observador. Son características concretas y su valor depende de ellas mismas o

de ninguna otra.

En un sistema informático, la observación u obtención de datos de una entidad interna significa el uso de las APIs (*Application Programming Interface*) para obtener sus valores en tiempo real o el uso de sensores en el caso de entidades externas. En el caso de las entidades externas, prácticamente todas las que se usan en un sistema software son representadas en el sistema de alguna manera.

La mayor dificultad para el soporte del Awareness es identificar los elementos de los que se requiere Awareness.

Para identificar estos elementos los desarrolladores pueden utilizar alguna de las siguientes técnicas:

- El diseño orientado al Awareness de Situación [Endsley et al., 2003].
- Los marcos descriptivos de varios tipos de Awareness específicos (Capítulo 3 Sección 4).
- La experiencia adquirida en desarrollos anteriores y expresada en tipos de Awareness concretos modelados previamente para dichos sistemas.

El enfoque utilizado en este trabajo es el de analizar los requerimientos de información dinámica de los usuarios durante la ejecución de tareas. El usuario utilizará dicha información actualizada para tomar decisiones durante la tarea, de tal forma que cumpla los objetivos previstos. Este enfoque es aplicable a cualquier metodología y se basa en el papel del Awareness en la toma y ejecución de decisiones.

Además de los elementos que conforman un tipo de Awareness, identificamos los Elementos Compuestos (*awCompositeElement*, Figura 45), los cuales representan elementos creados por el propio entendimiento del observador, que en este caso es el propio software.

En el caso de los seres humanos, el entendimiento produce conocimiento generado a través de procesos mentales utilizando información previa como la ex-

periencia y los valores de los elementos percibidos. En el caso de un sistema informático, un proceso de entendimiento. implica operar los valores capturados apoyándose en procesos u operaciones previamente definidas, ya sean algoritmos matemáticos o técnicas inteligentes. Este proceso creará nuevos elementos a nivel de comprensión, que aunque está limitada por los mecanismos implementados, no deja de ser útil.

Este nuevo conocimiento facilita niveles superiores de entendimiento a los observadores de la información de Awareness, los cuales pueden ahorrarse el tiempo y la energía que gastarían en procesar dichos datos. No obstante, en algunos casos puede ser útil obligar al usuario a procesar dichos valores, como es el caso de los entornos de aprendizaje. No obstante, el nivel de comprensión permite responder más rápido a situaciones complejas que pueden generar situaciones de riesgo, como por ejemplo, la conducción de vehículos, la gestión de emergencias, etc.

Los elementos compuestos utilizan el valor de otros elementos (ya sean concretos o compuestos) para definir su propio valor.

Como componente final del Awareness encontramos las Proyecciones de Awareness (*awProjection*, Figura 47). Una proyección es el cálculo del valor de los elementos en un momento del tiempo futuro. Estas proyecciones generan un nuevo conjunto de valores para los elementos del Awareness, pudiendo utilizar una o varios valores de dichos elementos a lo largo del tiempo. Para esto se utiliza lo que llamamos una función de proyección, la cual calcula nuevos valores para los elementos en un momento de tiempo futuro.

Las proyecciones permiten crear planes a futuro, aunque siempre debe contemplarse el factor de la inexactitud de las proyecciones, ya que no son exactas ni contundentes.

Estos componentes reflejan la estructura del Awareness de Situación presentada por [Endsley, 1995], la cual identifica los elementos del entorno (*awConcre-*

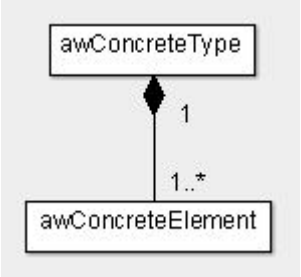


Figura 43. Elementos concretos del Awareness. Similares a los atributos de una clase.

teElement), su entendimiento en conjunto (*awCompositeElement*) y la capacidad de proyectar sus valores en el tiempo (*awProjection*).

Para el observador, un tipo de Awareness es visto como un estado mental se forma a partir de recibir constantemente un conjunto de valores de ciertos elementos del entorno, lo cual significa que para el sistema un tipo de Awareness representa el conjunto de datos que debe proporcionarle al observador a través de las interfaces de usuario o las APIs de comunicación para agentes artificiales.

En los siguientes puntos se ejemplificará cada componente del modelo conceptual del Awareness a partir de los tipos de Awareness encontrados en la literatura científica.

3.1.1. *Elementos concretos del Awareness: Nivel de Percepción*

Los elementos concretos (*awConcreteElement*, Figura 43)del Awareness representan los atributos o propiedades que el observador (en este caso, el sistema software) puede capturar directamente de la entidad observada usando los propios medios sensoriales (incluida la API software de comunicación) del observador.

Supongamos que durante una tarea determinada se requiere:

- El Awareness de la localización física de los usuarios.
- El Awareness de la localización de sus punteros en el área de trabajo colaborativo.

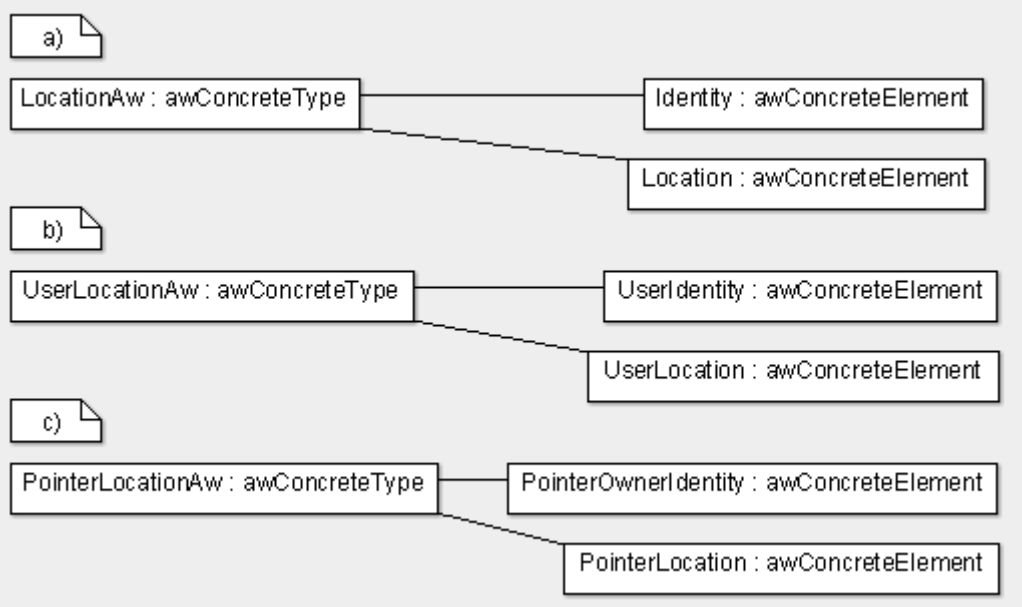


Figura 44. **a)** Modelo del *Location Awareness*. **b)** Awareness de Localización de los Usuarios. **c)** Awareness de Localización de los punteros.

El elemento común de los dos requerimientos anteriores es la "localización". El Awareness de Localización (Capítulo 3 Sección 4.2) cubre perfectamente la descripción general de lo que se requiere, pero en éste caso específico, hay que crear dos tipos distintos de Awareness: uno para la localización de los usuarios y otro para la localización de los punteros.

En el entorno de ejemplo, la localización física de los usuarios puede ser descrita por las propiedades Latitud, Longitud. En otro entorno puede ser descrita por la Calle y el Número. Pueden utilizarse más o menos atributos para describir la localización. Sin embargo, nuestro elemento concreto es la Localización y el Awareness obtenido va a ser el Awareness de Localización.

Para representar la localización de los punteros pueden utilizarse las propiedades X,Y de la entidad Puntero, obtenidas de los clientes software de cada usuario. En otro entorno se podrían utilizar otras propiedades, pero el uso sería el mismo: conocer la localización del puntero de cada usuario.

En este ejemplo los elementos concretos identificables son la localización y la

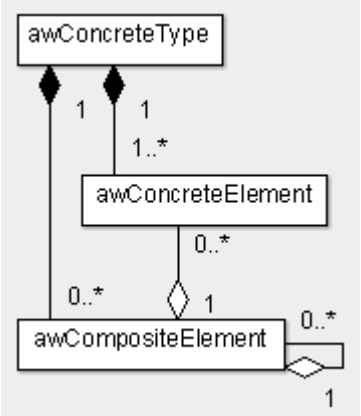


Figura 45. Elementos compuestos del Awareness. El valor de un elemento compuesto depende de otros elementos del Awareness, ya sean concretos o compuestos.

identidad de la entidad objetivo.

3.1.2. *Elementos compuestos del Awareness: Nivel de Comprensión*

El nivel de comprensión del Awareness se refiere a entender el significado de varios elementos concretos (no necesariamente del mismo tipo de Awareness) como un conjunto, de tal forma que se pueden generar nuevos elementos dependientes de los valores de otros elementos, ya sean concretos o compuestos. Es decir, un elemento compuesto (*awCompositeElement*, Figura 43) está formado por la combinación de los valores de varios elementos, lo cual sitúa al elemento compuesto en un nivel más alto de abstracción de información.

El elemento compuesto es similar a las propiedades o atributos derivados de una clase, cuyo valor depende del valor de otros atributos; por ejemplo, el nombre completo de una persona es resultado de concatenar sus nombres con sus apellidos. El nombre completo es un atributo derivado y los nombres y apellidos son atributos directos o concretos.

Como segundo ejemplo, supongamos que somos un alumno en un sistema de aprendizaje colaborativo. Queremos conocer el tiempo en el que cada compañero del grupo comienza una práctica en la cual puede recibir ayuda de los demás

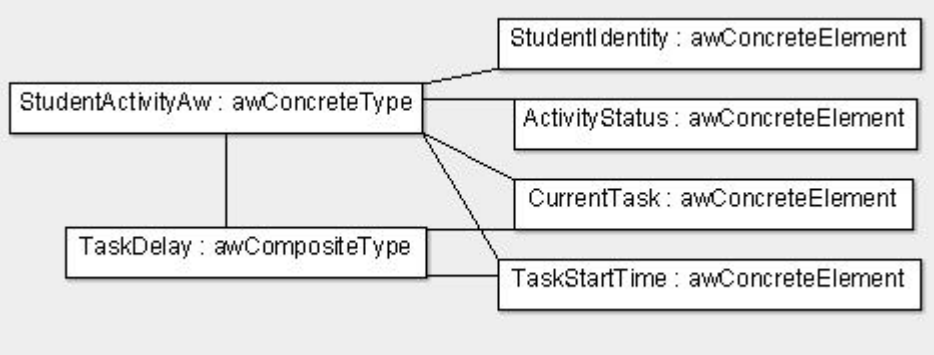


Figura 46. Awareness de Tardanza en una tarea.

alumnos. Si solo utilizamos los datos que podemos percibir tenemos la hora de inicio de las tareas y los alumnos ligados a ellas. Para que el sistema soporte el nivel de comprensión debe implementar mecanismos más elaborados que utilicen la información disponible.

Una forma de comprensión en este entorno es que el sistema muestre a través de alguna señal que un usuario con más de cierto tiempo desarrollando una tarea tiene dificultades. Por ejemplo, un color distintivo, una bandera o icono diferente, algo que les permita a los demás alumnos saber de forma rápida qué compañero necesita ayuda. Esto lo pueden saber si ven la hora de inicio de las tareas de cada compañero, pero eso los distraería de su propia tarea. Un vistazo rápido a una señal distintiva puede darles ese conocimiento de manera rápida.

Los elementos concretos que podemos identificar en el ejemplo anterior se muestran en la Figura 46, los cuales pueden provenir de distintas entidades de acuerdo a la estructura del dominio. El elemento compuesto *TaskDelay* representa la tardanza desde el comienzo de la tarea actual. Este elemento depende de la tarea actual y del tiempo de inicio. También puede utilizar otros valores provenientes de mecanismos de adaptación o personalización.

El sistema puede soportar el nivel de comprensión (elementos compuestos) de cualquier tipo de Awareness y facilitárselo a los usuarios. No obstante, distintas formas de pensar podrían conducir a distintos elementos compuestos derivados

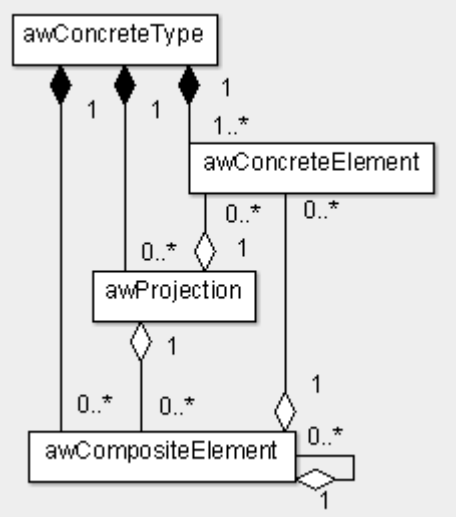


Figura 47. La proyección como elemento del Awareness

de las mismas necesidades de información. Por esta razón, el diseño cuidadoso de los mecanismos de Awareness para el soporte de los niveles superiores debe hacerse como parte de cualquier metodología en la que se incluya el soporte del Awareness.

3.1.3. *Proyecciones: Nivel de Proyección del Awareness*

El Awareness se forma en un individuo a partir de recibir constantemente información de ciertos elementos del entorno, es decir, una sucesión de datos en tiempo presente. Una proyección (en lo que respecta al Awareness) representa los valores de dichos elementos del Awareness en el futuro.

La proyección responde a la pregunta de "*¿qué valores tendrán estos elementos de Awareness en X tiempo?*". Visualmente, una proyección puede verse como una gráfica en función del tiempo, en la cual cada punto de la ecuación representa los valores de los elementos del Awareness representados en dicha proyección.

La utilidad de las proyecciones de Awareness radica en que permiten planificar acciones o decisiones con antelación a su utilización.

Como ejemplo de las proyecciones que se utilizan comúnmente por el grueso de

la población se encuentran las previsiones del estado del tiempo, las previsiones de tráfico, tiempos de llegada durante el transporte a otros lugares, etc. Otras proyecciones más específicas permiten conocer en aproximación el tiempo que va a tardar en aterrizar un avión o la cantidad de tiempo que va a poder seguir volando con el combustible restante.

El soporte de las proyecciones del Awareness se basa en funciones que pueden requerir analizar los datos de Awareness capturados u observados previamente. Por esta razón puede ser necesario guardar los datos de Awareness observados, aunque esto signifique agregar al desarrollo su representación, manipulación y mantenimiento. Sin embargo, los beneficios de las proyecciones de Awareness hacen que su soporte sea prácticamente necesario.

La forma de calcular las proyecciones es flexible, siempre y cuando devuelva instancias del propio tipo de Awareness y utilice un momento en el futuro como parámetro para hacer el cálculo. Si fuera un momento en el pasado, entonces no sería una proyección, sino una extracción de datos sobre la memoria o el repositorio de las instancias pasadas del tipo de Awareness.

A modo de ejemplo, la Figura 48 se muestra una representación gráfica del Awareness de Ancho de Banda (*Bandwidth Awareness*). Cada punto es el valor del único elemento concreto que representa el valor del ancho de banda en ese momento en el tiempo. El Awareness se proporciona señalando el momento actual (*Current time*) y los valores proyectados de este elemento (la proyección está compuesta por este único elemento concreto del Awareness) se señalan de un color distinto a los valores pasados del mismo elemento concreto.

Una vez descritos los elementos del modelo del Awareness, describimos a continuación el proceso para caracterizar un tipo de Awareness a un dominio específico.

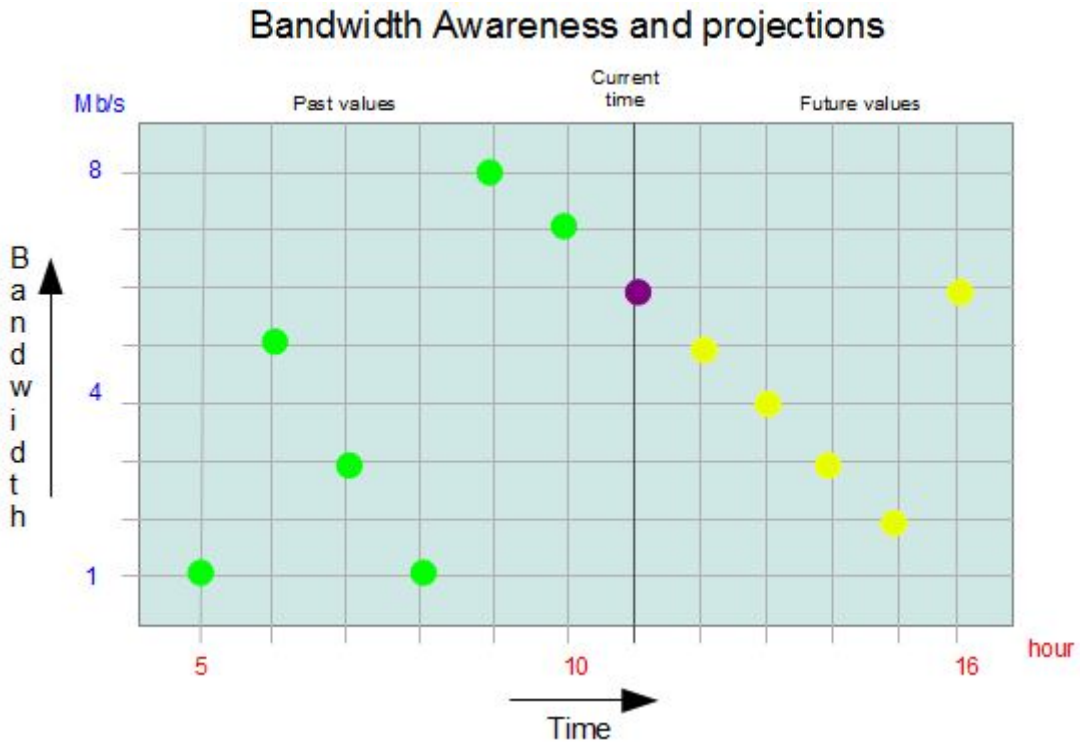


Figura 48. Awareness del ancho de banda y varias proyecciones del mismo (Creado para el ejemplo).

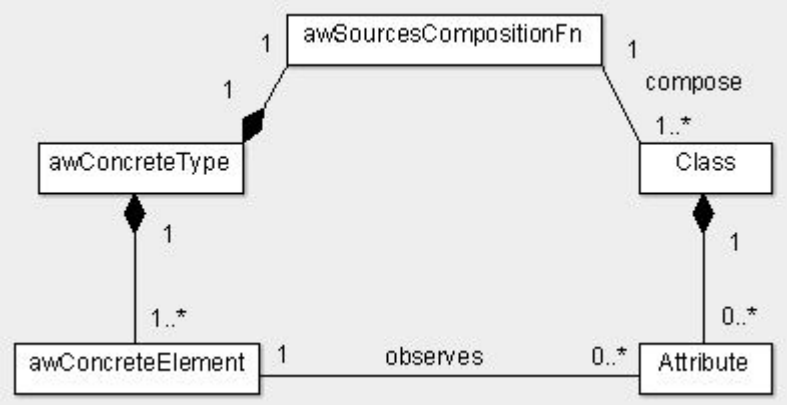


Figura 49. Relaciones para caracterizar un tipo de Awarenses con una o varias clases del dominio.

3.2. Método para la Caracterización del Awareness

El proceso de caracterización del Awareness tiene como propósito relacionar un tipo de Awareness con una o varias entidades del dominio o del sistema (clases o tablas, por ejemplo) para el cual se está desarrollando el software. Este proceso es relativamente sencillo una vez que se dispone de un modelo apropiado para representar cualquier tipo de Awareness, como es el caso de los modelos presentados en este trabajo.

La Figura 49 las relaciones entre los distintos elementos del modelado de un dominio y las entidades que componen el modelo del Awareness. Un tipo de Awareness (*awConcreteType*) se caracteriza a través de enlazar sus elementos concretos (*awConcreteElement*) con atributos (*attribute*) de las clases del dominio (*Class*) a las cuales representa.

Por ejemplo, el Awareness *UserPresenceAw* (representación conceptual del Awareness de Presencia del Usuario) se caracteriza enlazando los elementos concretos *username* e *isonline* con los atributos *name* de la clase *User* y el atributo *active_session* de la clase *Session*. Para componer dichas clases y que puedan obtenerse instancias concretas y unificadas de las mismas se utiliza la entidad *awSourcesCompositionFn*.

El tipo de Awareness no representa una nueva entidad en el dominio (aunque pudiera manipularse como tal), sino una forma de estructurar y ordenar los datos de una determinada entidad, de tal forma que puedan ser manipulados y transmitidos a los actores del sistema como datos de Awareness.

Una relación de uno a uno entre el elemento *awConcreteElement* y el *Attribute* (de una clase) es suficiente para definir que un determinado elemento concreto observa o toma su valor de un determinado atributo de una clase, vista o clase derivada [Balsters, 2003].

Las relaciones y entidades mostradas en la Figura 49 son suficientes para caracterizar un tipo de Awareness con una o varias entidades del sistema o del dominio. Sin embargo, para terminar de definir el tipo de Awareness es necesario definir los elementos compuestos y las proyecciones, en caso de que el tipo de Awareness lo requiera.

La separación entre la definición de los tipos de Awareness frente a la definición de las clases o tablas tiene un sentido organizacional. Es decir, definir los tipos de Awareness por separado permite darles una nomenclatura diferente y más apropiada, la cual puede ser diferente a la nomenclatura usada para las demás entidades del sistema.

Soportar la definición del Awareness por separado también puede ser útil en el caso de mantener una memoria o bitácora de los datos de Awareness ligados al momento en el cual se capturaron. Ésto con el fin de soportar proyecciones complejas que requieran dichos datos históricos. De esta forma, las clases del sistema no son sobrecargadas con dicho proceso y dependiendo de las técnicas utilizadas se puede separar este servicio para dar más flexibilidad y potencia a los mecanismos de soporte del Awareness.

3.3. Consideraciones técnicas

El modelado del Awareness llena una carencia muy importante en lo que respecta al soporte correcto y completo del Awareness. Sin embargo, existen

algunas dificultades técnicas que probablemente han sido una de las principales causas para optar por un soporte del Awareness incompleto pero probablemente más manejable y factible, dada la poca base teórica y conceptual disponible y localizable hasta ahora.

Identificamos dos aspectos técnicos que complican el soporte del Awareness a nivel de manejo de datos y caracterización. El primer aspecto es que los tipos de Awareness deben estar ligados a una sola clase (o clase derivada), tabla o vista. El segundo aspecto es el soporte de las proyecciones a nivel de datos.

A continuación describimos cada una de ellas.

3.3.1. *Caracterización del Awareness a través de múltiples clases del dominio*

Un tipo de Awareness puede caracterizarse en un dominio usando más de una clase del dominio o del sistema (Figura 50). Por ejemplo, para mostrar los usuarios en línea se puede definir el Awareness de Presencia caracterizándolo con atributos de la clase Usuario y de la clase Sesion. Dichas tablas están relacionadas y pueden unificarse para obtener de ellas las instancias requeridas por el Awareness de Presencia.

El proceso de unificación de fuentes de datos para el Awareness, las cuales son las clases del dominio, se representa con la entidad *awSourcesCompositionFn* y puede generar una clase derivada intermedia, una vista de datos, etc.

Ciertos procesos encontrados en el *Data Warehouse* (DWH) [Kimball and Ross, 2002] son muy similares en objetivos y en técnicas al proceso de modelado y caracterización del Awareness. El DWH se utiliza para obtener una visión global pero focalizada del estado de un sistema con el fin de apoyar la toma de decisiones. Este proceso puede verse como un Awareness global del sistema, aunque ciertamente el Awareness se maneja también a menor escala para mejorar la interacción con el sistema y/o con los demás usuarios.

Creemos que es importante entender esta similitud ya que el DWH propor-

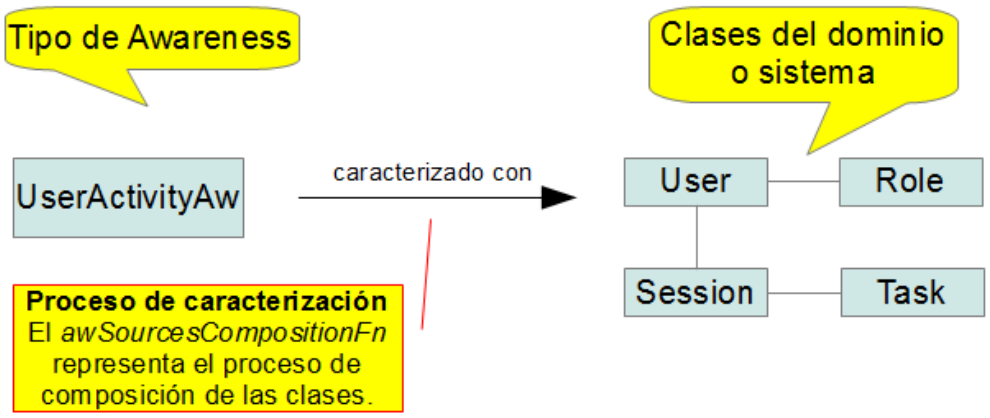


Figura 50. Diagrama de la caracterización del Awareness y de la composición de sus fuentes de datos.

ciona técnicas maduras y probadas en lo que respecta a integración, extracción y manipulación de datos, lo cual es necesario para el soporte del Awareness, ya sea a mayor o menos escala.

3.3.2. *Soporte de las proyecciones de Awareness*

Como hemos mencionado antes, el cálculo de las proyecciones puede depender de los valores historicos de los datos de Awareness. Esta situación debe ser tomada en cuenta durante el desarrollo de las herramientas y técnicas para dar soporte al Awareness. Esta situación no es exclusiva al conjunto de modelos que presentamos, sino que está presente en todas las formas de soporte al Awareness en general, siendo un problema poco tratado a nivel científico, ya que su principal repercusión es técnica.

Los datos de Awareness están ligados al tiempo por definición. Por lo tanto, ya sea para soportar proyecciones de Awareness o el más sencillo tipo de Awareness, es necesario tomar en cuenta el tiempo en el que se capturan los datos de Awareness y el cambio de sus valores.

4. Modelo de Requerimiento de Awareness

Cada aspecto y característica funcional y no funcional de un software parte de un conjunto de requerimientos y objetivos a cumplir. Las tareas y actividades posibles en las que participan entidades internas y externas están definidas por el dominio de aplicación, el cual, a su vez, está delimitado por los requerimientos de los usuarios y del propio sistema.

El Awareness es un estado mental requerido por los usuarios para la toma de decisiones y para la interacción con el sistema y con otros usuarios. Ya que el Awareness es un requerimiento de los usuarios entonces debe tratarse como tal, desde las primeras fases del desarrollo hasta el mantenimiento de todos los mecanismos implementados para su soporte.

El Awareness raramente ha sido tratado como requerimiento. Sin embargo, como hemos visto en el Capítulo 3 Sección 5.2, el Awareness ya está siendo aceptado como requerimiento fundamental en los sistemas colaborativos, aunque tiene presencia implícita en la gran mayoría de los sistemas informáticos actuales.

La representación de los requerimientos también ha cambiado. Las metodologías de desarrollo tradicionales usualmente utilizan los requerimientos como guía externa para desarrollar el sistema. Cuando cambian o se descubren nuevos requerimientos (proceso iterativo) se actualizan por separado el software, los requerimientos y la documentación sobre cada uno de ellos. Esta situación es conocida por los desarrolladores y debido a ello se han creado nuevas formas de desarrollo, que aunque no son tan ampliamente utilizadas comienzan a sustituir las formas actuales de desarrollar software.

El Desarrollo Basado en Modelos (MDD) [Mellor et al., 2003] es una de éstas técnicas en las cuales se aprovechan los requerimientos para generar partes del sistema y documentación, de tal forma que el cambio en los requerimientos no afecta en gran medida al esfuerzo en el desarrollo del software, ya que el sistema es generado a partir de los requerimientos y restricciones representadas usando

modelos descriptivos.

Actualmente el MDD sigue siendo poco utilizado por el común denominador de las empresas medianas y pequeñas. Sin embargo, la necesidad de modelar los requerimientos y el propio dominio de trabajo se ve reflejada en los esfuerzos para integrar la especificación de requerimientos y restricciones en los marcos de trabajo como Ruby On Rails[2], Django[3], entre otros.

El MDD depende de la capacidad de modelar requerimientos, tareas, interfaces de usuario, entre otros componentes comúnes en los sistemas software. Por tanto, a fin de representar los requerimientos de Awareness, un modelo de requerimientos se vuelve necesario. El requerimiento nos permite entender el sistema, los motivos para su desarrollo y la justificación de cada necesidad (requerimiento) que tienen los usuarios y el propio sistema.

Un aspecto importante en el modelado de los requerimientos de software es que permiten que el sistema enlace los requerimientos con los mecanismos para su soporte. Ésto es de gran importancia a la hora de validar el sistema a través de los requerimientos que soporta, a la vez que permite guiar el desarrollo si se priorizan los requerimientos más importantes.

Nuestra propuesta para el modelado de los requerimientos de Awareness está basada en el marco de referencia del soporte del Awareness presentado en la Sección 2. Este modelo se ha desarrollado con el propósito de servir como documentación y como modelo descriptivo del requerimiento del Awareness en un software, siendo perfectamente extendible y adaptable a las herramientas utilizadas durante el desarrollo y ejecución del sistema.

4.1. Representacion del Requerimiento de Awareness

Un requerimiento de Awareness representa la necesidad de información actualizada que tiene un grupo de actores del sistema durante una situación específica.

[2] `www.rubyonrails.org`
[3] `www.djangoproject.com`

Los requerimientos del usuario rara vez son obtenidos a la primera durante el desarrollo. Más bien, estos requerimientos se van descubriendo durante el proceso iterativo de desarrollo y pruebas con el usuario mediante prototipos.

Un requerimiento de información actualizada o Awareness debe expresar todo lo necesario para saber que información requiren los usuarios, quiénes exactamente, cuándo, cómo, dónde y porqué. De esta forma, un requerimiento de Awareness representa una necesidad que debe cubrir el sistema. Por otra parte, un requerimiento que desaparece, representa un conjunto de procesos y mecanismos que ya no son necesarios en el sistema y que deben removerse.

La Tabla 13 muestra el modelo del Requerimiento de Awareness expresado como una plantilla, la cual está basada en el marco de referencia presentado en la Sección 2 y que se interpreta textualmente de la siguiente manera:

"*El requerimiento de Awareness con identificador EJEMPLOREQ presenta la necesidad de los usuarios USUARIOS en la tarea TAREA de tener Awareness de la información INFORMACION, en caso de cumplirse las condiciones CONDICIONES. Las preferencias de interacción de éste requerimiento son PREFERENCIAS_INTERACCION y la justificación para su soporte es JUSTIFICACION. Otros comentarios son: OTROS.*"

Un ejemplo del uso de la plantilla propuesta se muestra en la Tabla 14, la cual se interpreta como:

"*El bombero experto requiere Awareness del nombre, cantidad de gasolina, nivel de agua y nombre del conductor de todos los camiones de bomberos que no estén en reparación, durante toda la tarea de Gestión de camiones durante una emergencia, siendo la información visible en su dispositivo móvil o en las pantallas grandes del centro de control.* ".

El modelo de requerimientos de Awareness presentado en esta sección no pretende ser específico a una plataforma o metodología de desarrollo. Por el contrario, busca la adaptabilidad a nuevos entornos y formas distintas de desa-

Plantilla de Requerimiento de Awareness	
Id	*EJEMPLOREQ* (Identificador.)
Usuario (Quién)	USUARIO (Usuario o grupo con requerimiento de Awareness.)
Tarea (Dónde)	TAREA (Tarea en la que se requiere el Awareness.)
Información (Qué)	INFORMACION (Infomación de Awareness requerida. Su estructura general y las condiciones que delimitan las instancias que la van a componer.)
Aplicabilidad (Cuándo)	CONDICIONES (Si se cumplen las condiciones se proporciona la información de Awareness.)
Interacción(Cómo)	PREFERENCIAS_INTERACCION (La forma en la que se deben presentar la información de Awareness y los medios preferidos.)
Justificación (Por qué)	JUSTIFICACION
Otros	OTROS

Tabla 13. Plantilla para representar un Requerimiento de Awareness en lenguaje natural. Puede relacionarse directamente con uno o varios modelos para el soporte del Awareness.

Requerimiento de Awareness	
Id	*BomberoExpertoReq1*
Usuario	El bombero experto.
Tarea	Gestión de camiónes de bomberos durante emergencias.
Información	Toda la información de los cambiones de bomberos (nombre, cantidad de gasolina, nivel de agua, conductor) excepto de los camiones en reparación.
Aplicabilidad	Durante toda la tarea.
Justificación	Durante una emergencia se necesita conocer el estado de los camiones a fin de planificar el uso de cada uno de ellos.
Otros	Interfaces especiales para pantallas grandes (centro de control) pero también para los dispositivos móviles utilizados por los bomberos.

Tabla 14. Ejemplo de un Requerimiento de Awareness

rrollo, siendo más una guía conceptual orientada al desarrollo que un formato de representación específico e inamovible.

5. Ejemplo general

La descripción de modelos conceptuales es una tarea difícilmente asimilable por el lector si no es acompañada de varios ejemplos de aplicación sobre los cuales basarse para explicar los conceptos, entidades y relaciones en los que intervienen los modelos descritos.

Como ejemplo general de esta sección utilizaremos el proceso de desarrollo de un sistema colaborativo de apoyo a los desarrollos de software libre distribuido y soportado por sistemas de control de versiones de código. Le llamaremos CRCS como las siglas de *Collaborative Review Control System*.

El CRCS va a ser utilizado para proveer un entorno de trabajo síncrono y asíncrono a los desarrolladores de software que utilizan sistemas de control de versiones (RCS) y que se encuentran distanciados físicamente, lo cual es algo muy común en el mundo del software libre. Su objetivo no es reemplazar el modelo de trabajo del desarrollo de software distribuido usando sistemas como Git[4], Subversion[5], Mercurial[6], entre otros[7], sino el de agilizarlo y proporcionar otros canales más elaborados para la colaboración entre desarrolladores.

El ciclo normal de trabajo, una vez que se está en un proyecto y se tienen permisos de escritura en el repositorio principal del RCS utilizado es el siguiente:

- Crear copia de trabajo local o clonar el repositorio principal.
- Realizar cambios.
- Integrar cambios de otros usuarios.

[4] `http://git-scm.com`
[5] `http://subversion.tigris.org`
[6] `http://mercurial.selenic.com`
[7] `http://en.wikipedia.org/wiki/Comparison_of_revision_control_software`

- Resolver conflictos en el repositorio local.
- Guardar cambios en repositorio local.
- Enviar cambios al repositorio principal para sincronizarlo.
- Realizar cambios y volver a empezar el proceso.

Este ciclo de trabajo incluye el coordinarse con otros desarrolladores para saber en qué áreas trabajar o para resolver conflictos de código por haber cambiado el mismo código de forma distinta.

En ocaciones se requiere compartir código entre varios desarrolladores a fin de analizarlo, cambiarlo y darle el visto bueno como parte del código fuente del proyecto. Este es un proceso muy tardado o ineficiente si se utilizan medios de comunicación tradicionales (y disponibles) en estos casos como es el correo electrónico o el chat. Por supuesto, estos medios son importantes, pero se requieren mejores técnicas de interacción para agilizar este proceso de revisión colaborativa de código.

Además de proporcionar un Awareness general del estado del desarrollo del proyecto a los usuarios, el CRCS pretende proveer un espacio de trabajo colaborativo para dicha situación, además de sentar las bases para los encuentros informales [Gutwin et al., 2008] con desarrolladores de otros proyectos durante estas revisiones, ya que su experiencia puede ser de gran utilidad.

Podemos entonces identificar un espacio de trabajo compartido con capacidad para promover encuentros informales en tiempo real. Además, la realización de las revisiones de código debe promover la interacción asíncrona entre los usuarios, dado que no siempre pueden estar presentes en tiempo real pero es importante que conozcan el trabajo realizado por otros usuarios.

A continuación describiremos algunas de las tareas y entidades del sistema CRCS a fin de proporcionar un entendimiento más detallado del mismo.

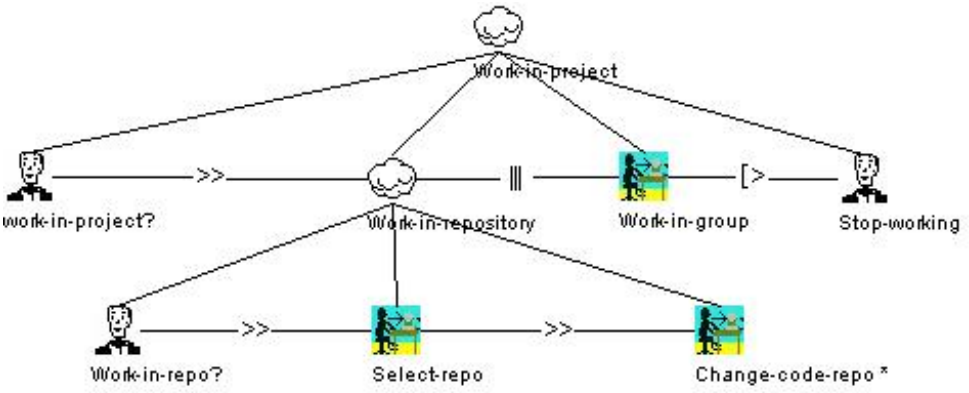


Figura 51. Estructura de tareas sobre el proceso inicial de trabajar en un proyecto de software.

5.1. Tareas del sistema CRCS

Existen diversas maneras de representar la estructura de tareas de un sistema software, algunas de las cuales han sido descritas en la Sección 4.3. Para este caso particular utilizaremos la notación CTT (Concur Task Trees), la cual nos permite describir el ejemplo a distintos niveles de abstracción, detallando o generalizando según se requiera.

A grandes rasgos, un desarrollador implicado en un proyecto de software libre normalmente utilizará un sistema de control de versiones para trabajar en el código o documentación del proyecto. Puede estar implicado en otros proyectos, por lo cual el flujo inicial de tareas es similar al mostrado en la Figura 51. Una vez que el desarrollador decide trabajar en un proyecto puede empezar a trabajar con el repositorio de código a la vez que empieza una sesión de trabajo en grupo, usando en este caso el sistema CRCS. El hecho de poder realizar las dos labores al mismo tiempo es bastante normal, ya que puede estar cambiando archivos y código a la vez que mantiene simultáneamente conversaciones y revisiones con otros desarrolladores.

Cabe recordar que el propósito del sistema CRCS no es cambiar el entorno de

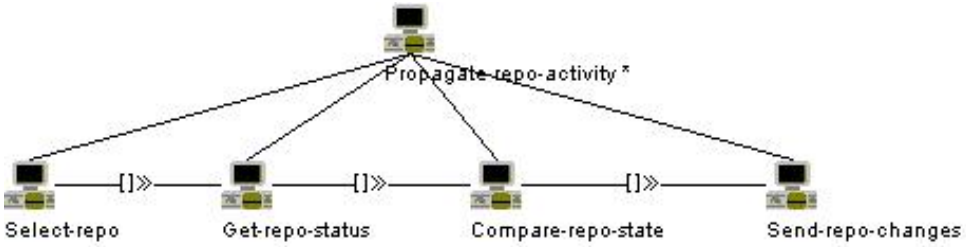


Figura 52. Estructura de tareas sobre el proceso de propagación de cambios en un repositorio local realizado por el agente de software local.

trabajo del desarrollador, el cual puede ser un IDE como Eclipse[8] o un conjunto de herramientas a su gusto (Vim[9], Emacs[10], etc.), sino proporcionar un entorno para el trabajo en grupo y para contribuir al Awareness global sobre el proyecto y sus colaboradores.

El sistema CRCS es independiente a los repositorios de código individuales de cada usuario. Por esta razón el CRCS requiere un agente local para suministrarle la información de todos los cambios realizados por cada usuario a sus correspondientes repositorios locales, los cuales reflejan su propio trabajo en tiempo real y cuya información alimenta al propio sistema.

El flujo de tareas del agente local se muestra en al Figura 52. Aquí se muestra como el agente local propaga al sistema CRCS toda la actividad del desarrollador sobre un repositorio de código determinado, permitiendo al desarrollador controlar la información que es compartida con los demás desarrolladores, aunque todos los cambios irán finalmente en el código del proyecto. El agente selecciona el repositorio de entre los repositorios monitorizados, revisa su estado en comparación con el último capturado. Si hay cambios envía toda la información correspondiente al CRCS y actualiza el estado del repositorio para la siguiente revisión de actividades. Así permanece siempre que esté activado.

[8] `http://www.eclipse.org`
[9] `http://www.vim.org`
[10] `http://www.gnu.org/s/emacs/`

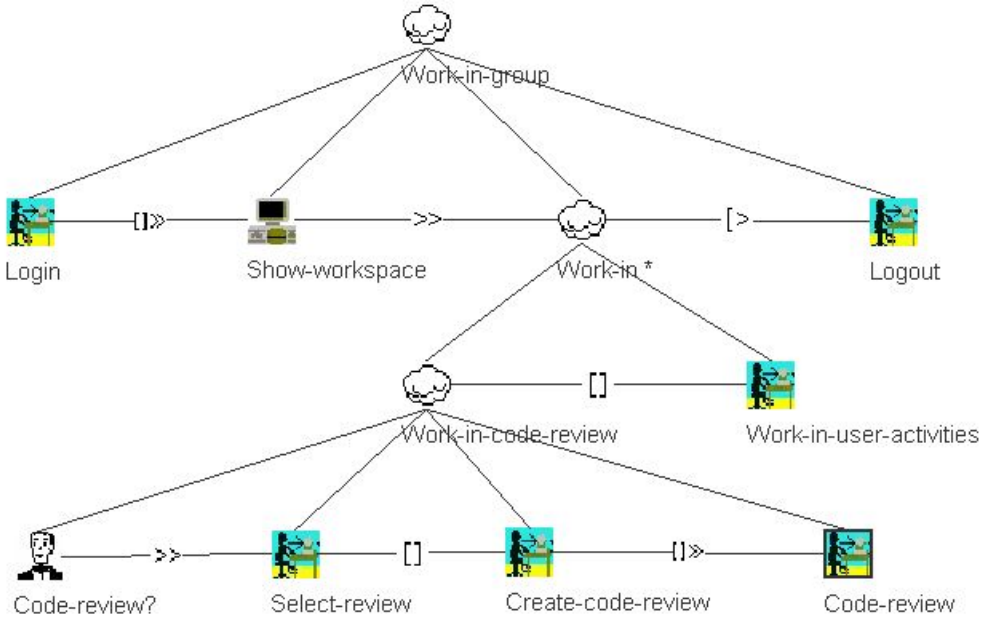


Figura 53. Estructura general de tareas del sistem CRCS. Muy familiar a otras. Inicio de sesión, subtareas del sistema y fin de sesión. Solo se muestra el sub árbol para trabajar con revisiones de código, siendo éste el proceso que nos interesa por ser colaborativo y tener varios requerimientos de información actualizada.

El diagrama general del sistema CRCS se muestra en la Figura 53 y presenta una forma muy general y utilzada que va desde el inicio de sesión a la utilización del espacio de trabajo compartido para realizar diversas tareas, de las cuales nos interesa la de `Work-in-code-review` (trabajar en revisión de código), la cual engloba otras subtareas pero principalmente identifica un proceso colaborativo en el que varios usuarios del sistema revisan y cambian un código de forma conjunta y en tiempo real, teniendo cada uno ciertos roles que les permiten hacer o no determinadas acciones.

La tarea en la que se realiza el proceso de revisar código de forma colaborativa tiene como nombre `Code-review`. La Figura 55 muestra el diagrama colaborativo CTT de dicha tarea. Cada sub tarea mostrada en este diagrama va a ser ejecutada para todos los usuarios que accedan a la tarea principal sin importar el rol que tengan. Según el rol de cada usuario durante la revisión tendrá más o menos opciones para hacer, además de tener más o menos elementos informativos en la

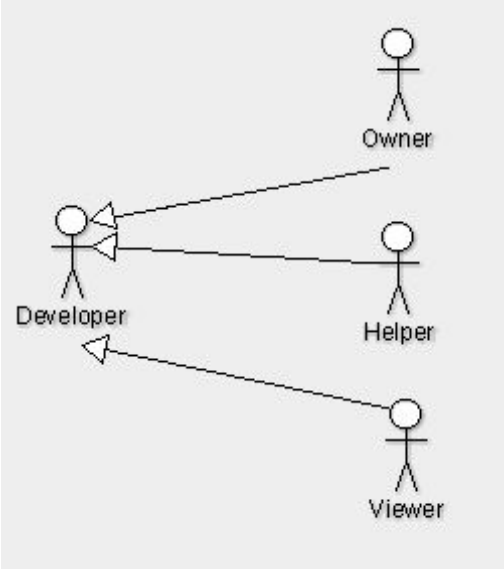


Figura 54. Roles del *Developer* en una Revisión de código: *Owner*, *Helper* y *Viewer*.

interfaz de usuario.

Esta tarea colaborativa tiene una estructura descrita por el patrón de comunicación `Debate moderado` [Isla Montes, 2007], en el cual se elige un moderador que va a controlar el turno de palabra. En este caso el moderador es el creador de la revisión y lo que se controla no es la palabra (ya que todos pueden comunicarse por mensajes para no cortar el flujo de comunicación) sino la capacidad de escritura del código revisado, el cual, por razones tecnológicas y de control, sólo debe ser editado por un desarrollador a la vez, ya que este proceso se hace en línea y el código editado reside en el sistema CRCS como parte del contexto compartido para dicha revisión de código.

La Figura 54 se muestran los roles del usuario *Developer* para la revisión de código. El usuario es el *Owner* de una revisión cuando la revisión ha sido creada por el mismo usuario. Un usuario *Helper* en una revisión puede realizar cambios en una revisión y puede tomarse del rol que tiene dicho usuario en el proyecto al que pertenece la revisión (son los mismos roles). El usuario *Viewer* no puede realizar cambios en una revisión de código, pero puede interactuar con los otros

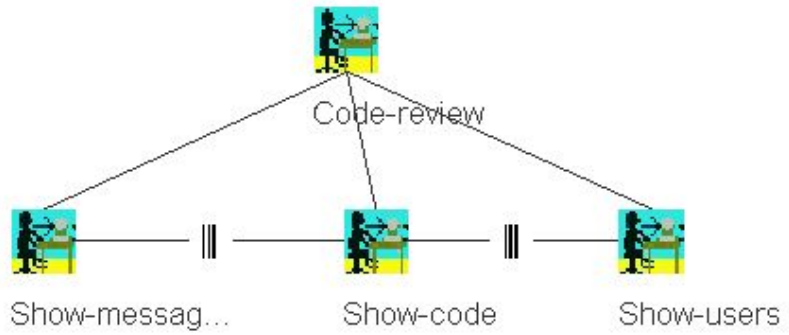


Figura 55. Diagrama colaborativo de la tarea Code Review o Revisión de Código.

desarrolladores en la revisión.

Un usuario puede ser el *Owner* de una revisión de código ser *Viewer* en otra revisión.

La Figura 56 muestra las subtareas de la tarea `Code-review` para el rol *Owner*, el cual, como dijimos antes, es el encargado de moderar la revisión, por lo cual requiere más información que el resto de los usuarios, ya que debe realizar más actividades durante esta tarea. Estas actividades extra se encuentran bajo la tarea abstracta `Manage-review`, la cual incluye el manejo de usuarios (sacar o invitar a la revisión) y el manejo del permiso de escritura.

Tres tareas muestran la información principal que se requiere proporcionar durante dicha tarea: Mostrar Código, Mostrar Mensajes y Mostrar Usuarios. Esta información no es la única requerida como veremos más adelante.

Puede parecer que quién crea una revisión de código carga con demasiado trabajo como para estar atento al código como propósito. Sin embargo, esto puede manejarse pidiendo a otro colaborador que tome el rol de *owner* para que así, el usuario que realmente necesita estar concentrado en el código pueda hacerlo.

La información sobre el código en cuestión puede ser muy importante para los otros desarrolladores. Por ejemplo, saber quien ha modificado ese código, archivo

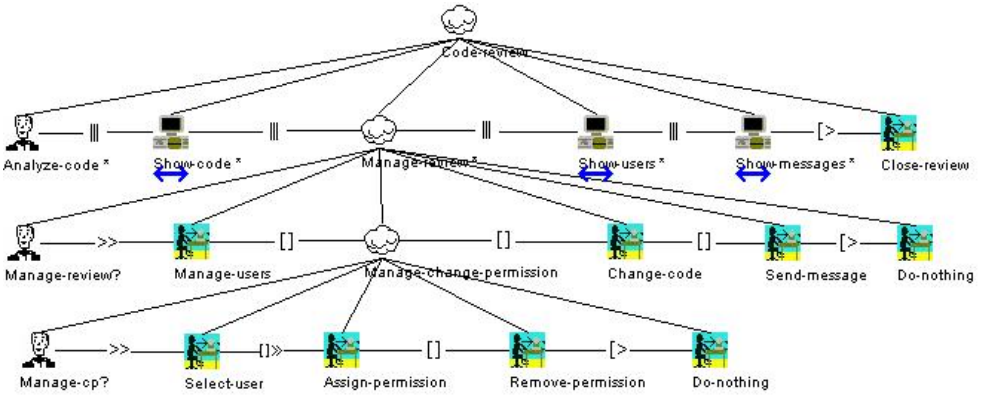


Figura 56. Tareas para el rol *Owner*.

o sección previamente, de tal forma que los demás sepan quienes están familiarizados con él. Esta información es específica para este dominio de aplicación y no está documentada como un tipo de Awareness concreto. Sin embargo, como podemos ver, las necesidades de información van apareciendo conforme se analizan las decisiones a tomar y los beneficios grupales durante la tarea realizada.

La Figura 57 muestra la misma tarea `Code-review` para los usuarios que entren con el rol de *helper* o ayudante, los cuales pueden cambiar el código (una vez dado el permiso), pero no pueden cerrar la revisión ni asignar el permiso de escritura. Su función es participar activamente en la revisión.

La Figura 58 muestra las actividades a realizar por los usuarios que entren a la revisión con el rol de *viewer*, lo cual significa que su participación está restringida al canal común el cual puede ser un chat, pero sin poder cambiar el código, ya sea por falta de experiencia o porque ya hay suficientes *helpers* con más conocimientos o mayor tiempo en la revisión.

El objetivo de la revisión colaborativa es combinar la experiencia e ideas de varios desarrolladores a fin de resolver un problema importante, mejorar substancialmente un código o simplemente compartir puntos de vista y aumentar el conocimiento general del grupo de trabajo.

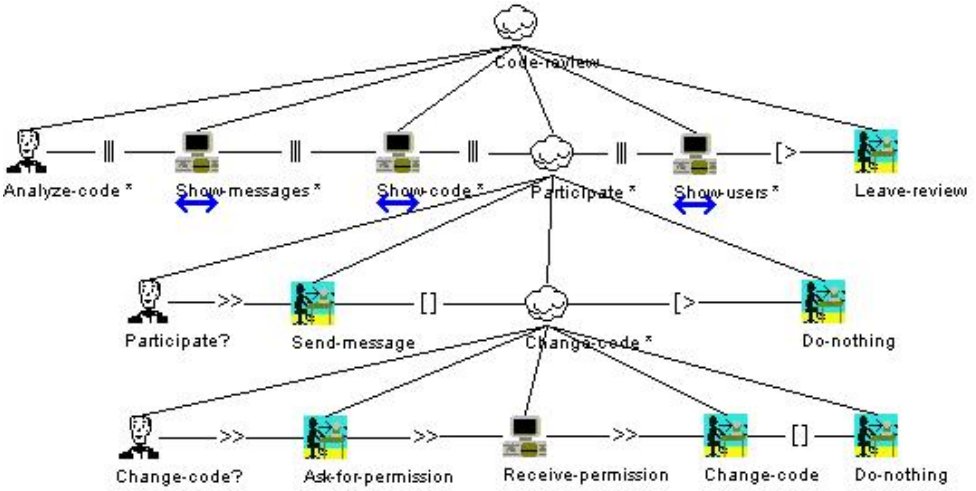


Figura 57. Tareas para el rol *Helper*.

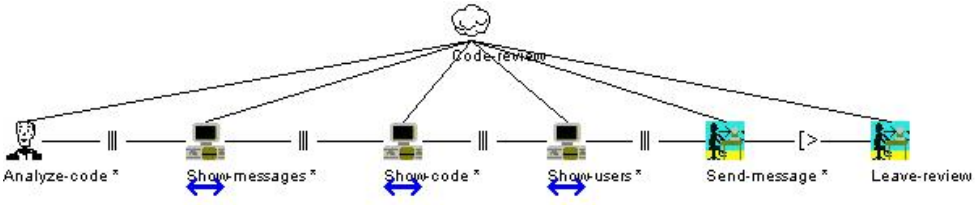


Figura 58. Tareas para el rol Viewer.

Esta tarea colaborativa puede analizarse desde varias perspectivas y según los objetivos que busquemos encontraremos distintos requerimientos de información por parte de los usuarios para cumplir dichos objetivos. Sin embargo, el proceso de extracción de estos requerimientos depende en gran medida de las pruebas reales (tal vez con prototipos) y de la retroalimentación recibida de los usuarios después de utilizar el sistma.

Entonces, a fin de soportar los requerimientos específicos de información debemos contar con modelos genéricos para el soporte del Awareness pero usables para el desarrollo de software, los cuales puedan irse refinando conforme avanza el desarrollo sin que produzcan más desventajas que ventajas a la hora de integrar cambios.

A continuación presentamos un análisis general de las entidades identificadas para el sistema CRCS, sabiendo a ciencia cierta que puede haber más de las que a continuación presentamos.

5.2. Entidades del dominio

En práticamente todos los sistemas se manejan de alguna forma los usuarios, sus respectivos roles o permisos y las entidades que permiten manipular el dominio del propio sistema. Por ejemplo, un sistema de facturación manejará productos, pólizas, tal vez almacenes, entre otras cosas.

Fuera del dominio existen entidades igualmente importantes, como es la sesión. Los sistemas colaborativos pueden incluir una amplia variedad de entidades para representar y manipular la estructura del propio sistema a través de los mismos medios con los que se manipulan las entidades del dominio.

Por ejemplo, un sistema colaborativo puede manejar las entidades usuario, role, grupo, tarea, sesión, objetivo, actividad, etc. [Gutiérrez et al., 2006]. Estas entidades se utilizan para representar y manipular el propio sistema y en las metodologías basadas en modelos, pueden usarse para generar el código de la propia aplicación.

Para el sistema CRCS, identificamos la siguientes entidades:

Developer. Representa a un desarrollador de software. En este caso los desarrolladores son los usuarios del sistema. Un desarrollador puede participar en varios proyectos a la vez.

Project. Representa un proyecto de software, en el cual estan implicados uno o varios desarrolladores y colaboradores que no codifican pero que pueden documentar, probar, buscar errores o compartir ideas. Un desarrolladore puede tener tres roles en un proyecto: Owner (dueño), Commiter (desarrollador), Viewer (colaborador).

Activity. Representa la actividad en el repositorio del proyecto propagado de los repositorios locales de cada desarrollador al sistema a través del agente local. Puede representar cualquier cambio como agregar un archivo, modificar, eliminar, etc.

Review. Representa una revisión de código en la que pueden participar varios desarrolladores. Puede estar relacionada con un cambio y contener varios mensajes entre sus desarrolladores, entendiendo que el mismo mensaje se va a enviar a todos los usuarios en la revisión.

Message. Representa un mensaje entre dos desarrolladores. Los mensajes pueden estar relacionados con un proyecto, un cambio o una revisión.

La Figura 78 muestra el diagrama de clases del dominio CRCS. El diagrama no se muestra las instancias propias del sistema, como son las Tareas, la Sesión, entre otras. Un modelo conceptual adecuado para los sistemas colaborativos puede verse en [Gutiérrez et al., 2006]. Las entidades del sistema pueden enlazarse con entidades del dominio, pero eso dependerá de los requerimientos del sistema.

En las Tablas 15, 16, 17, 18 y 19 se muestran los atributos de cada entidad. De esta forma será posible describir el proceso de caracterización del Awareness. Las Tablas 20, 21 and 22 muestras la estructura de las clases que representan las relaciones de muchos a muchos de las clases referenciadas.

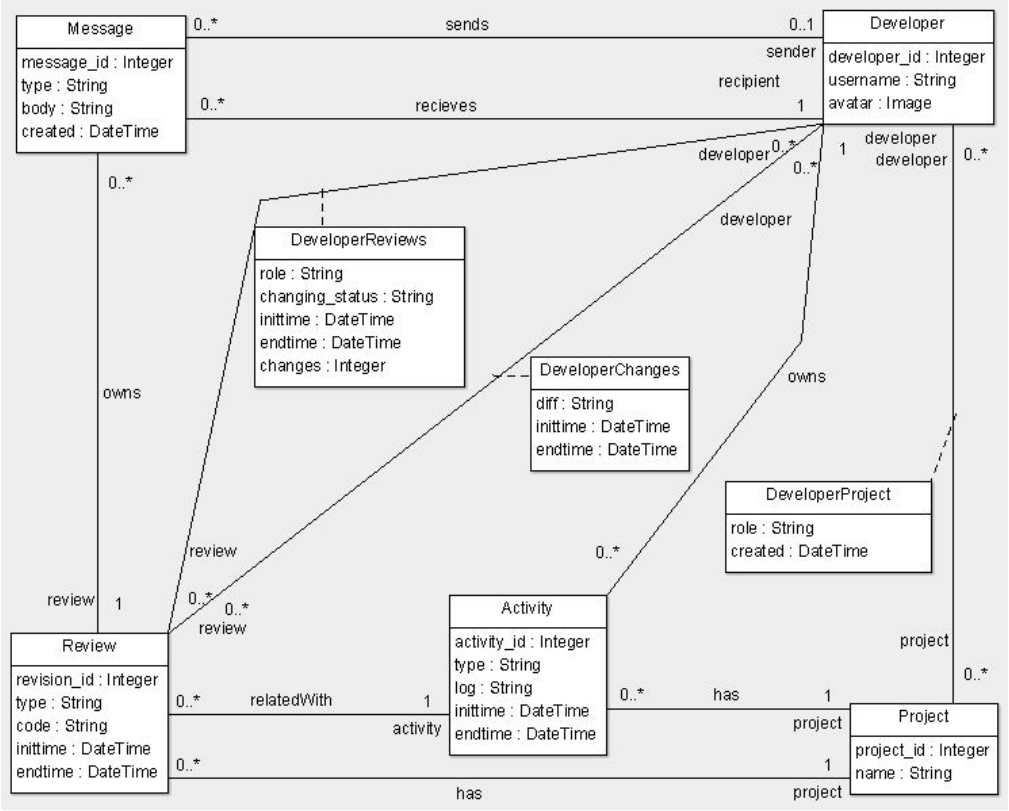


Figura 59. Diagrama de clases de las entidades del dominio CRCS.

Clase Developer		
Atributo	**Tipo**	**Descripción**
developer_id	Integer	Identificador principal.
username	String	Nombre de usuario único.
avatar	Image	Imagen de avatar.

Tabla 15. Clase *Developer*. Representa a los desarrolladores usuarios del sistema.

Clase Project		
Atributo	**Tipo**	**Descripción**
project_id	Integer	Identificador único.
owner	Developer	Desarrollador creador del proyecto.
name	String	Nombre del projecto.

Tabla 16. Clase *Project*. Representa a los proyectos de desarrollo de software.

Clase Activity		
Atributo	**Tipo**	**Descripción**
activity_id	Integer	Identificador único.
owner	Developer	Desarrollador que realiza la actividad.
type	String	Puede adquirir los siguientes valores: Commit, Working.
log	String	Bitácora de los cambios hechos, ya sean en un envío (Commit) o durante el trabajo intermedio (Working).
inittime	DateTime	Inicio de la actividad o del Commit.
endtime	DateTime	Finalización de la actividad.

Tabla 17. Clase *Activity*. Representa todas las activadades posibles del proyecto.

Clase Review		
Atributo	**Tipo**	**Descripción**
revision_id	Integer	Identificador único.
owner	Developer	Desarrollador que creó la revisión.
type	String	Tipo de revisión: Código, Cambios, Archivos.
code	String	Código en revisión.
inittime	DateTime	Fecha y hora de iniciación.
endtime	DateTime	Fecha y hora de terminación.

Tabla 18. Clase *Review*. Representa las revisiones colaborativas. En este ejemplo nos enfocaremos a las revisiones de código solamente.

Clase Message		
Atributo	**Tipo**	**Descripción**
message_id	Integer	Identificador único.
revision	Review	Revisión a la que pertenece o nulo si no pertenece a ninguna.
sender	Developer	Desarrollador que envía el mensaje.
recipient	Developer	Desarrolladore destinatario o nulo si es público en una revisión.
type	String	Puede ser Public o Private.
body	String	Cuerpo del mensaje.
created	DateType	Momento en el que se envió el mensaje.

Tabla 19. Clase *Message*. Representa los mensajes enviados entre desarrolladores y que pueden estar dentro de una revisión de código.

Clase DeveloperProjects		
Atributo	**Tipo**	**Descripción**
developerprojects_id	Integer	Identificador único.
developer	Developer	Usuario que participa o posee un proyecto.
project	Project	Proyecto en el que participa el usuario.
role	String	Puede ser `Owner` (dueño), `Member` (miembro) y `Viewer` (miembro que no puede hacer cambios).
created	DateTime	Fecha de creación del proyecto o de alta como miembro o visitante en otro.

Tabla 20. Clase *DeveloperProjects*. Representa los proyectos en los que participa un desarrollador, ya sea como creador del proyecto o como miembro del mismo.

Clase DeveloperReviews		
Atributo	**Tipo**	**Descripción**
developer	Developer	Desarrollador en la revisión.
revision	Review	Revisión en la que participa el desarrollador.
role	String	Puede ser `Owner` si el desarrollador es el creador de la revisión, `Helper` si es un miembro activo del proyecto o `Viewer` si es un miembro pasivo del proyecto el cual no puede hacer cambios en las revisiones de código.
changing_status	String	Puede ser `None`, `Asking` o `Changing`. Dice si el usuario está pidiendo hacer cambios o no, o si está haciendo cambios.
inittime	DateTime	Momento en el que el desarrollador crea la revisión o cuando se une a ella.
endtime	DateTime	Momento en el que el desarrollador cierra o sale de la revisión.
changes	Integer	Cantidad de cambios realizados por el desarrollador en esta revisión. El sistema incrementa este número cada vez que el desarrollador realiza un cambio.

Tabla 21. Clase *DeveloperReviews*. Representa las revisiones en las que participa un desarrollador, ya sea como creador de la revisión o como participante activo o pasivo.

Clase DeveloperChanges		
Atributo	**Tipo**	**Descripción**
developer	Developer	Desarrollador que realiza el cambio.
revision	Review	Revisión en la que sucede el cambio.
inittime	DateTime	Momento en el que se inicia un cambio.
endtime	DateTime	Momento en el que finaliza un cambio. Puede ser nulo en caso de que el cambio no haya finalizado.
diff	String	Cambio representado en formado Diff (`http://en.wikipedia.org/wiki/Diff`) tipo texto.

Tabla 22. Clase *DeveloperChanges*. Representa los cambios hechos en una revisión colaborativa.

Una vez descrita de forma general la estructura de tareas y las entidades del dominio podemos iniciar la definición de los requerimientos de Awareness.

5.3. Requerimientos de Awareness

A cada requerimento le antecede un análisis de la tarea en la que se ha detectado dicho requerimiento y en la que el Awareness juega un papel importante en las decisiones tomadas y en la interacción con el sistema o con los demás usuarios.

El método "*Situation Awareness oriented design*" [Endsley, 2001] da una guía para analizar los requerimientos de Awareness de Situación durante una tarea o actividad. No está orientado específicamente a la extracción de requisitos, pero ayuda a localizar las decisiones importantes que se deben tomar durante la tarea y a través de esas decisiones, ayuda a detectar los requerimientos de información de Awareness.

A continuación describimos algunos requerimentos de Awareness relacionados con la tarea colaborativa `Code-review`, la cual representa a la tarea de revisión de código colaborativa. Esta tarea puede tener más requerimentos de Awareness. Sin embargo, para no hacer más complejo y largo el ejemplo utilizaremos tres requerimentos de Awareness en los cuales se requiere proporcionar los tres niveles del Awareness: Percepción, Comprensión y Proyección.

La Tabla 23 describe un Requerimiento de Awareness aplicable a todos los desarrolladores que entran en una revisión colaborativa de código, ya sea como *Owner*, *Helper* o *Viewer*. La descripción del requerimiento es informal y desencadenará varios mecanismos de interacción y es una guía básica para el diseño de las interfaces de usuario. La definición de los tipos de Awareness y su caracterización permitirán enlazar modelos o especificaciones de más bajo nivel a estos requerimientos de alto nivel.

La Tabla 24 representa un Requerimiento de Awareness específico para los usuarios *Owner* y *Helper*, los cuales son quienes pueden cambiar el código durante la revisión.

La Tabla 25 muestra un requerimiento de Awareness específico para los desarrolladores que crean la revisión y que por lo tanto deben administrarla durante toda su duración. Este requerimiento debe aplicarse según las restricciones definidas en el mismo y sin afectar la aplicación de los otros requerimentos que afectan a este usuario. Así mismo este requeriiento extra se traducirá en más información transmitida para el usuario con rol *owner* durante la revisión colaborativa.

Una vez descritos los requerimientos de Awareness y presentada la estructura de clases del dominio CRCS podemos describir de forma más precisa la información de Awareness que va a ser transmitida a los distintos usuarios dependiendo de sus necesidades.

Los tipos de Awareness requeridos por los usuarios se modelan de acuerdo a las necesidades de información descritas en los requerimientos de Awareness.

Para representar de los tipos de Awareness encontrados se utilizarán los modelos del Awareness propuestos anteriormente en Sección 3.

5.4. Tipos de Awareness

En base a los requerimientos de Awareness definidos anteriormente podemos proseguir definiendo los tipos de Awareness que representan los datos requeridos del sistema para mantener conciencia del entorno de interés, el cual en este caso

Requerimiento de Awareness	
Id	*code-review-general-awreq*
Usuario	Todos los usuarios en la revisión sin importar su rol en la misma.
Tarea	`Code-Review`
Información	* De la revisión, el desarrollador que la creó, el tiempo de inicio, la duración hasta el momento actual y los desarrolladores participantes. * De los desarrolladores participantes, su nombre de usuario, avatar y, bajo demanda (mostrar solo cuando el usuario quiera), los proyectos en los que participa. También se necesita saber su rol en la revisión. * Los mensajes públicos en la revisión los privados destinados o enviados por cada usuario. * El código en revisión actualizado a los últimos cambios.
Aplicabilidad	Proporcionar **siempre** la información de toda la revisión excepto la que es bajo demanda.
Interacción	Agrupar datos de la revisión. Separar panel de usuarios con acceso a información detallada. Los mensajes y los usuarios deben poder esconderse dejando todo el espacio al código.
Justificación	De los usuarios es necesario saber sus datos principales, su rol para saber quien maneja la revisión y quien puede hacer cambios. De la revisión es importante saber quien la lleva y cuanto tiempo lleva abierta. Los participantes en ella son muy importantes. Si hay muchas participaciones se incrementa el aprendizaje y se puede mejorar bastante el código. El código y los mensajes actualizados deben estár disponibles para todos.
Otros	No sobrecargar la interfaz, ya que hay mucha información en la revisión. Aprovechar colores, formas e iconos para expresar datos de los usuarios, tiempos y mensajes. El código debe ocupar la mayor parte del espacio, luego los mensajes y por último la información de los usuarios.

Tabla 23. Requerimiento de Awareness de todos los usuarios que participan en una revisión de código colaborativa, independientemente de su rol.

Requerimiento de Awareness	
Id	*code-review-changes-aw*
Usuario	El *Owner* y los*Helper*s en la revisión.
Tarea	`Code-Review`
Información	* De los usuarios, la cantidad de cambios que han realizado, los usuarios que solicitan permisos de escritura y el usuario que está realizando cambios actualmente en el código en revisión.
Aplicabilidad	Mostrar el número de cambios y la petición de permisos durante toda la revisión. Mostrar el nombre del desarrollador que hace cambios solo cuando se estén realizando.
Interacción	Agregar esta información a la interfaz que muestra los usuarios en la revisión. Hacerla siempre visible cuando haga falta.
Justificación	El *owner* y los colaboradores necesitan saber quienes han editado mucho el código, de tal forma que puedan darle oportunidad a los otros colaboradores. Saber quien está realizando el cambio actual también es necesario.
Otros	

Tabla 24. Requerimiento de Awareness relacionado con los cambios al código compartido.

Requerimiento de Awareness	
Id	*code-review-owner-awreq*
Usuario	El *owner* de cada revisión.
Tarea	`Code-Review`
Información	* Del usuario que está realizando cambios, la duración de su turno de cambios en el código.
Aplicabilidad	Mostrar la duración actual de la operación solo cuando alguien esté cambiando el código.
Interacción	Poner estos datos en una pequeña UI junto al código, de tal forma que este usuario sepa que se está cambiando el código y la duración actual.
Justificación	La duración del cambio permite apresurar al desarrollador si lleva mucho tiempo haciendo cambios y evita que otros puedan hacer también cambios.
Otros	

Tabla 25. Requerimiento de Awareness relacionado con la duración del turno de cambios al código compartido. Si el usuario tarda mucho puede requerirse el cancelar dicho cambio.

tiene que ver con la revisión de código, los desarrolladores participantes y la comunicación escrita entre ellos.

Cada elemento de Awareness que es requerido forma parte o está relacionado con alguna entidad o concepto. Por ejemplo, la edad, el color de ojos, el color de piel, etc., pueden estar relacionados con un ser humano. Un tipo de Awareness con esos elementos puede llamarse Awareness del Aspecto físico. Los tipos de Awareness descritos se refieren a entidades del propio dominio del CRCS. De esta manera el proceso de especificación de los tipos de Awareness es algo parecido a especificar la estructura de una clase en el paradigma de orientación a objetos. Es decir, se agrupan los elementos de información que describan aspectos de una entidad en común, como una persona, un artefacto, un mecanismo, etc.

Describimos a continuación los tipos de Awareness necesarios para cubrir las necesidades de información descritas en los requerimientos de Awareness definidos previamente:

Awareness relacionados con la revisión

El Requerimiento de Awareness *code-review-general-awreq* plantea la necesidad de recibir información actualizada de la revisón en la que está participando un usuario. En este aspecto, varios aspectos de la revisión son de importancia para los desarrolladores.

Varios de esos elementos de información requeridos pueden agruparse en un tipo de Awareness, ya que se refieren al Awareness de una misma instancia: la revisión.

En base a los datos requeridos podemos definir el Awareness de Revisión o *ReviewAw* (Figura 60) a través de sus elementos concretos y elementos compuestos, los cuales concuerdan de cierta manera con los atributos de la clase *Review*.

La Tabla 26 describe los elementos de Awareness concretos y compuestos que conforman el Awareness de *Revisión*. Por cuestiones de familiaridad, los nombres

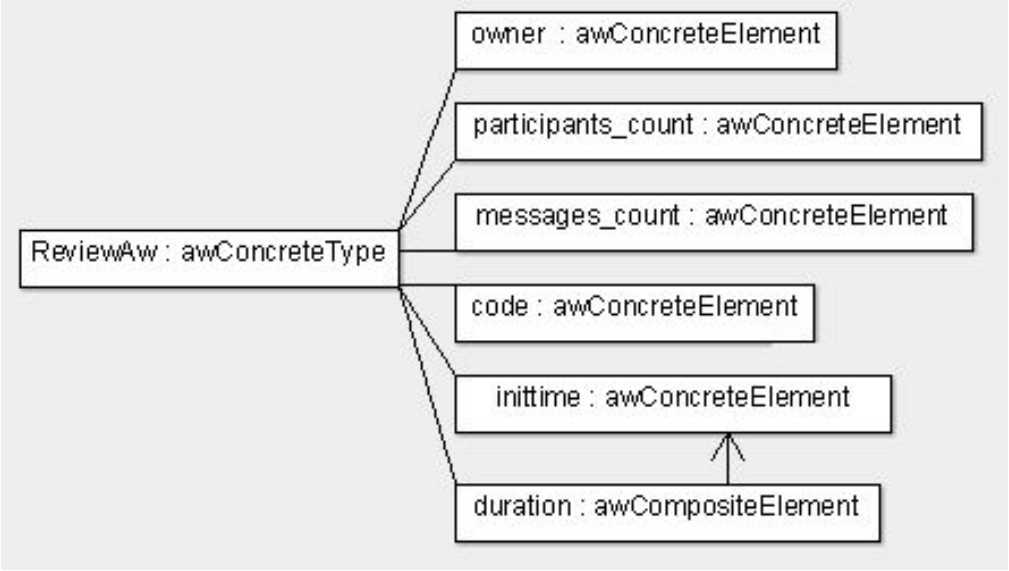


Figura 60. Awareness de la Revisión.

de los elementos concretos son iguales al nombre de los atributos de las clases del dominio de los cuales pueden obtenerse sus valores.

Los nombres de los elementos de Awareness pueden ser cuales quiera, mientras permitan identificar un elemento de información. En este caso, al haberse hecho la definición del dominio primero, es más simple nombrar a los elementos de Awareness con los nombres de atributos ya conocidos, aunque esto no es una regla.

De hecho, si se pretenden utilizar estos tipos de Awareness en otro dominio, probablemente tomen su valor de atributos con nombres distintos, por lo cual conviene utilizar nombres genéricos pero descriptivos como altura, distancia, peso, etc., y dejar que el propio dominio defina el tipo de datos y todas las demás restricciones sobre el atributo que se enlace con el elemento concreto de Awareness.

La Tabla 27 los elementos que componen el *DeveloperInRevAw* (Figura 61). Un elemento en particular es el de *projects*, el cual no existe por sí mismo como un atributo de alguna clase en el dominio. Este elemente ha sido creado por una

Tipo de Awareness *ReviewAw*	
Elementos concretos	
owner	Desarrollador que ha creado la revisión.
participants_count	El número actual de participantes en la revisión.
messages_count	El número actual de mensajes públicos que se han enviado.
inittime	El tiempo de inicio de la revisión.
code	El código compartido.
Elementos compuestos	
duration	Duración de la revisión. Compuesto por la hora actual menos la hora de inicio *inittime*.

Tabla 26. Representa el Awareness adquirido a través de la percepción de los elementos de la entidad *Review*.

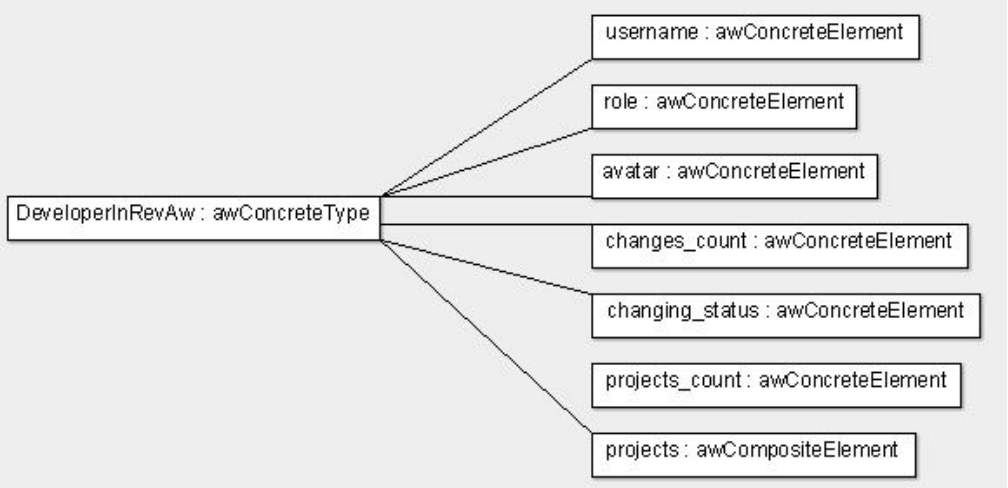


Figura 61. Awareness de la Revisión.

necesidad de información especial.

El valor de este elemento deberá ser calculado de alguna manera por el sistema. Este proceso de cálculo debe llevarse hasta la fase de diseño para que de alguna manera los desarrolladores especifiquen el dicho proceso de cálculo y lo liguen al propio elemento compuesto, de tal forma que se mantenga la estrutura de trazabilidad entre los mecanismos del sistema y los requerimentos que los justifican.

La Tabla 28 muestra los elementos de Awareness que lo componen (Figura

Tipo de Awareness *DeveloperInReviewAw*	
Elementos concretos	
name	Nombre de usuario del desarrollador.
role	Rol en la revisión, ya sea *Owner*, *Helper* o *Viewer*.
avatar	Avatar del usuario dependiendo de la modalidad de las interfaces de usuario.
changes_count	Número de veces que ha cambiado el código compartido.
changing_status	Si está cambiando o no el código.
projects_count	Número de proyectos en los que participa.
Elementos compuestos	
projects	Listado de los nombres de los proyectos en los que participa, distinguiendo los propios de los demás.

Tabla 27. Representa el Awareness sobre los datos de los desarrolladores dentro de una revisión. La estructura conceptual del *DeveloperInReviewAw* no se equipara a la de ninguna entidad del dominio, por lo cual sus datos se sacarán de varias entidades.

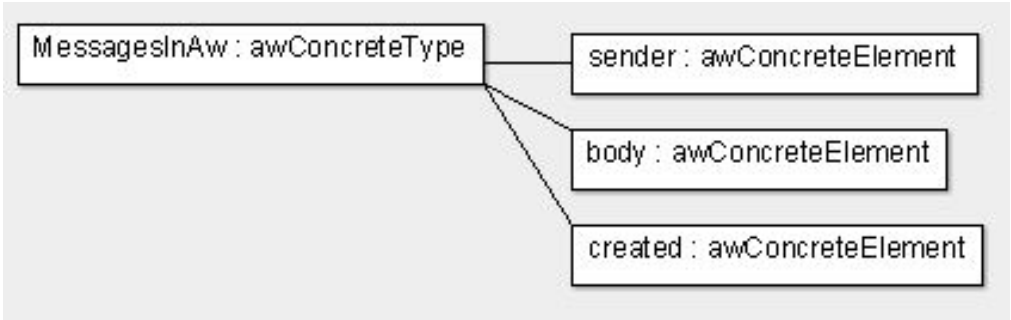


Figura 62. Awareness de la Revisión.

Tipo de Awareness *MessagesInRevAw*	
Elementos concretos	
sender	Nombre de usuario del desarrollador que envía el mensaje.
type	Tipo de mensaje (Público o privado).
body	Cuerpo del mensaje a ser mostrado.

Tabla 28. Awareness de los mensajes dentro de la revisión.

62). Los tipos de Awareness no especifican la forma en la que debe presentarse la información. Solamente especifican su estructura para después poder ligarla a los Requerimientos de Awareness y en base a ellos construir el soporte del Awareness hacia las capas superiores del desarrollo.

La Tabla 29 describe los elementos del tipo de Awareness *DevelopersAvailabilityAw* (Figure 63), el cual contiene un elemento de Awareness particular: una proyección.

La proyección *availability* funciona como un procedimiento que debe recibir un momento en el tiempo como entrada y que devuelve los valores de los elementos asociados a dicha proyección, los cuales en este caso son *inittime* y *endtime*.

El objetivo de esta proyección es dado un momento en el futuro, se pueda conocer la cantidad de desarrolladores cuyo periodo de revisión (entre el *inittime* y el *endtime*) cubre dicho periodo de tiempo. Dado que es un tiempo futuro, no se trata de una búsqueda histórica, ya que esos son datos pasados. Se necesita otro tipo de cálculos para obtener una estimación como la descrita. Es a lo que en este trabajo llamamos la función de proyección (*awProjectionFn*, Sección 3, Figura 42).

Los tipos de Awareness agrupan varios elementos que pueden estar presentes en diferentes Requerimientos de Awareness. Es decir, en un requerimiento puede estar presente el elemento *participants_ count* del tipo de Awareness *ReviewAw*, y en otro puede puede estar presente el elemento *messages_ count* del mismo *awConcreteType*.

Es necesario que existan mecanismos para ligar a una tarea el requerimiento

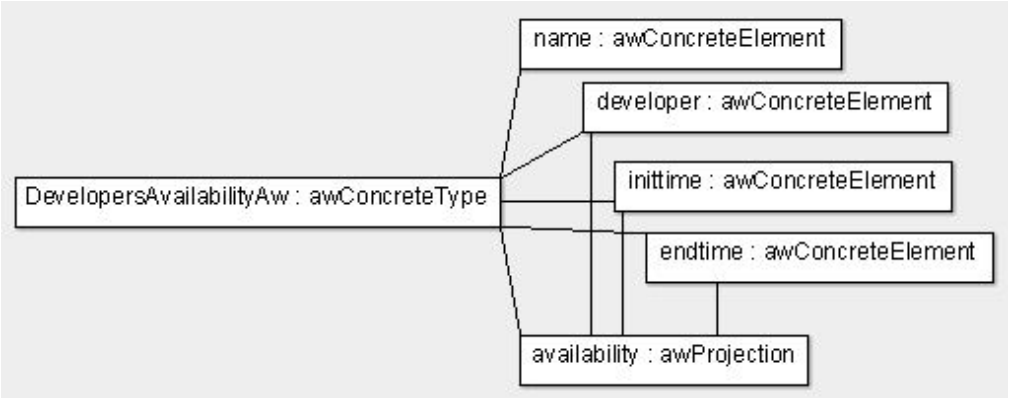


Figura 63. Awareness de la disponibilidad de los desarrolladores.

Tipo de Awareness *DevelopersAvailabilityAw*	
Elementos concretos	
name	Nombre del desarrollador que entró a una revisión.
inittime	Fecha y hora en la que el desarrollador se unió a una revisión.
endtime	Fecha y hora en la que el desarrollador salió de una revisión.
Proyecciones	
availability	Permite saber los momentos en el tiempo en los que probablemente estarán conectados los usuarios de interés para el desarrollador.

Tabla 29. Tipo de Awareness sobre la disponibilidad de los usuarios. Incluye una proyección que permite planificar la hora a la cual crear una revisión en base a la posibilidad de que más usuarios puedan estar presentes en el sistema.

de un conjunto específico de elementos del Awareness, sin tener que proporcionar todos los elementos que componen un tipo de Awareness. En el Capítulo 5 se muestra una manera de hacerlo adaptada a una metodología de desarrollo específica.

Una vez definidos los tipos de Awareness pueden caracterizarse o ligarse al dominio de trabajo, lo cual describimos en la siguiente sección.

5.5. Caracterización de los tipos de Awareness

La caracterización de los tipos de Awareness del ejemplo en dos pasos:

- Uniendo los elementos concretos de cada tipo de Awareness al atributo de la entidad del dominio o sistema de la cual obtendrá sus valores.
- Definiendo una función para componer los entidades a las cuales pertenecen los atributos enlazados anteriormente a los elementos concretos de Awareness. El propósito de esta función es que los datos manejados por el tipo de Awareness sean coherentes entre ellos y formen una estructura utilizable para obtener datos de ella.

En la Figura 64 se muestran los enlaces entre el tipo de Awareness *ReviewAw* y las entidades del dominio *Developer* y *Review*. Esta relación se obtiene indirectamente de las relaciones entre los atributos de dichas entidades y los elementos concretos del tipo de Awareness mencionado.

La función de composición de fuentes de datos llamada *ReviewAwCF* tiene como propśito enlazar las entidades *Developer* y *Review*, de tal forma que los datos manejados por el *ReviewAw* sean congruentes. Es decir, si se están utilizando los datos del `Review-00001`, el elemento *owner* va a apuntar al desarrollador que esté ligado a dicha revisión como el creador de la revisón. El cuidado de dicha congruencia o integridad relacional es responsabilidad de la función de composición de fuentes de datos de Awareness (*awSourcesCompositionFn*, Figura 49).

El resto de los tipos de Awareness ligados al dominio de trabajo se muestran en las Figuras 65, 66 y 67.

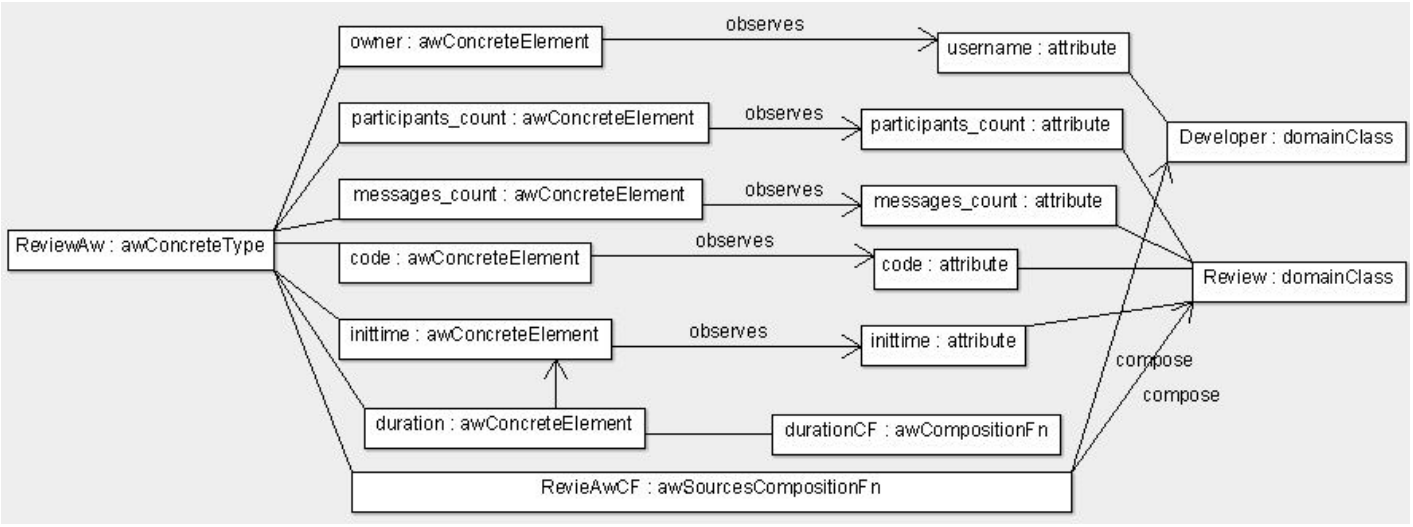


Figura 64. Caracterización del *ReviewAw*. Relacionado con las entidades *Developer* y *Review*.

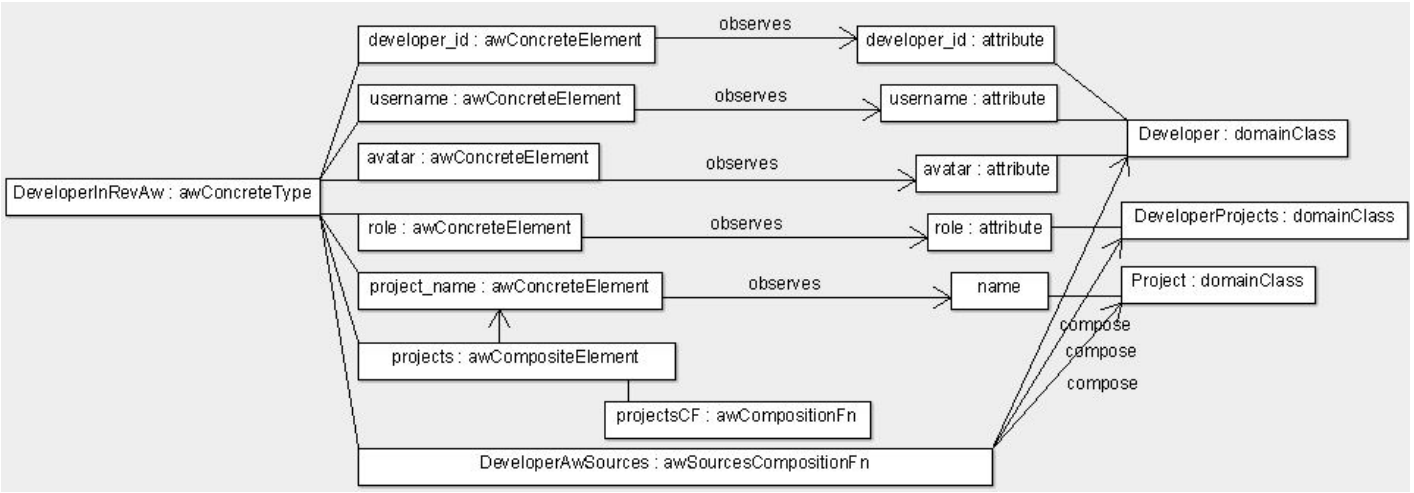


Figura 65. Caracterización del *DeveloperInRevAw*. Relacionado con las entidades *Developer*, *DeveloperProjects* y Project.

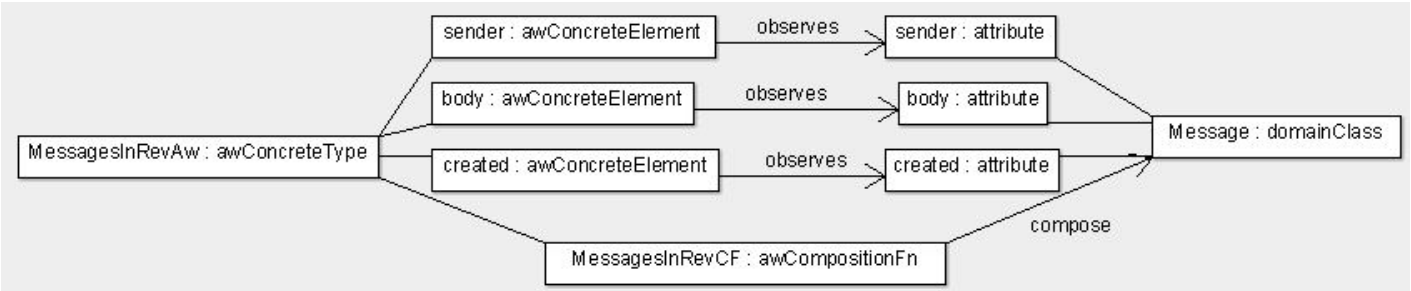


Figura 66. Caracterización del *MessageInRevAw*. Relacionado con la entidad *Message*.

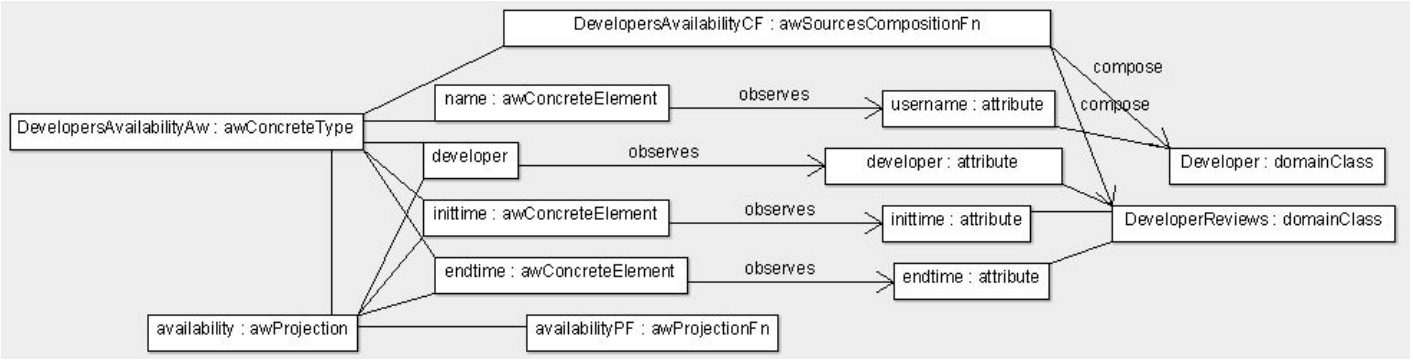


Figura 67. Caracterización del *DevelopersAvailabilityAw*. Relacionado con la entidad *Developer* y *DeveloperReviews*.

El propósito del ejemplo anterior es mostrar la capacidad de representacion del modelo del Awareness y el modelo de Requerimientos de Awareness.

Los requerimientos representados podrán ser utilizados en el diseño de las interfaces de usuario, no sin necesitar de otros parámetros para el control de la interfaz y del flujo de la interacción entre el sistema y el usuario o entre el usuario y los demás usuarios.

6. Conclusiones

Los modelos conceptuales desarrollados y el marco de referencia son resultado de unificar y generalizar las propuestas existentes para el soporte de varios tipos de Awareness distintos y usados en entornos muy diversos.

El marco de referencia se presenta a nivel conceptual, de tal forma que sea utilizable en múltiples metodologías a través de varios enfoques de desarrollo, como el desarrollo dirigido por modelos (MDD). Dicho marco es una caraterizacion general del awareness que permite definir las bases de su soporte a todos los niveles del desarrollo de un sistema.

El modelo del Awareness permite definir cualquier tipo de Awareness e instanciarlo (caracterización del Awareness) al dominio en el cual se esté trabajando.

Un tipo de Awareness consta de 3 componentes principales: el Elemento Concreto, el Elemento Compuesto y las Proyecciones, de acuerdo con la teoría del Awareness de Situación [Endsley, 1995], la cual tomamos como base teórica para el desarrollo de los modelos conceptuales propuestos.

Siguiendo nuestra propuesta, para definir un tipo de Awareness se describe primero un requerimiento de Awareness, en lenguaje natural usando el modelo propuesto en la Sección 4, en el cual se dislumbran los elementos de información requeridos y los atributos de las entidades del dominio que deben observarse o crearse (elementos compuestos o proyecciones) para proporcionar dicho Awareness. Una vez hecho ésto se procede a definir de manera más precisa el tipo de

Awareness a través de la especificación de sus elementos concretos, elementos compuestos y proyecciones.

Una vez definido un tipo de Awareness se procede a caracterizarlo usando algún tipo de enlace ("*mappings* " [Montero et al., 2006] por ejemplo) entre el tipo de Awareness y la clase del modelo de la que se quiera tener Awareness, y entre los elementos concretos y los atributos de dicha clase.

La definición de los tipos de Awareness permite detectar la necesidad de integrar y unificar fuentes de datos para proveer dicha información a través de las interfaces de usuario. En este aspecto, la teoría y las técnicas maduras del *Data Warehouse* pueden servir de apoyo para diseñar los procesos de obtención de datos de distintas fuentes.

Los desarrolladores deben diseñar e implementar los mecanismos técnicos para proporcionar a los actores del sistema los tipos de Awareness especificados en los requerimientos de acuerdo con las restricciones de cada uno de estos requerimientos.

Los modelos del Awareness han sido desarrollados, principalmente con miras a su integración dentro de una técnica de desarrollo dirigido por modelos, la cual puede aprovechar los modelos para automatizar partes del proceso de desarrollo, como puede ser la generación de código o la documentación.

Aspectos como el manejo de datos, la frecuencia de actualización, los lenguajes de representación, etc., corresponden a fases superiores del desarrollo y pueden ser muy diferentes dependiendo de las tecnologías disponibles y de las metodologías y herramientas de desarrollo utilizadas. Los modelos son la base para un soporte ordenado, más no un procedimiento fijo y específico para incluir el soporte del Awareness dentro alguna metodología.

Una ventaja de los modelos del Awareness es que sirven de guía durante el diseño de las entidades del dominio y durante la implementación de los procesos de transformación de datos. Por ejemplo, el detectar elementos compuestos y

proyecciones permite reconocer la necesidad de implementar los procesos, ya sean métodos u operaciones de clases que calculen dichos valores.

El objetivo del modelo de Requerimientos de Awareness es el de contemplar el Awareness para la toma de decisiones y la interacción desde las primeras fases del desarrollo, a fin de sentar las bases para un diseño e implementación que incluya el Awareness hasta las interfaces de usuario. De esta manera, la información proporcionada y todos los mecanismos asociados con este comportamiento tengan su justificación expresada en los requerimientos de Awareness, independientemente de la plataforma, tecnología o metodología usada para desarrollar y soportar el sistema de software.

Está claro que el MDD puede sacar mayor ventaja a los modelos del Awareness y de los requerimientos de Awareness, ya que uno de los objetivos del MDD es reaprovechar al máximo la especificación de los requerimientos y otros aspectos del sistema para generar código, interfaces de usuario y documentación. Sin embargo, no significa que las metodologías más tradicionales no se beneficien con estos modelos, ya que el hecho de conocer los aspectos que se deben soportar y documentar sobre los requerimientos de Awareness ya es una ventaja por sí misma.

Mientras más interactivo es un sistema, más mecanismos de Awareness requiere y más carga tendrá el sistema en lo que respecta al manejo y a la manipulación de los datos y entidades del dominio. Una buena detección y definición de las necesidades de Awareness permite trabajar en lo que realmente le permita al sistema cumplir sus objetivos a nivel de interacción, colaboracción, cooperación y comunicación.

CAPITULO 5

Extensión a UsiXML para el soporte del Awareness

1. Introducción

En los capítulos anteriores se han descrito de forma general los sistemas Groupware y el Awareness como aspecto esencial en la interacción y en la colaboración. Después hemos presentado un conjunto de modelos conceptuales que describen el Awareness y proporcionan las bases para su soporte controlado en el software.

En este capítulo presentamos la integración del Awareness a una metodología de desarrollo basada en modelos usando como base el marco de referencia del Awareness y los modelos descritos en el Capítulo 4.

Integrar un nuevo componente o extender cualquier metodología de desarrollo requiere cambiar parte del proceso de trabajo, extender los medios de representación y adaptar los procesos y las herramientas de soporte.

El integrar modelos conceptuales para la representación del Awareness implica cambiar los medios de definición de entidades y las relaciones entre varios de ellos. Además, en el caso de contar con procesos para manipular dichas entidades, es necesario adaptarlos para manipular los nuevos medios de definición, ya sean modelos o código.

En este trabajo hemos comenzado la integración del Awareness a nivel de representación extendiendo el lenguaje base de la metodología UsiXML.

Primeramente describiremos la metodología y su lenguaje de representación base. Después describiremos la entensión del lenguaje basada en los modelos del Awareness presentados. Posteriormente explicaremos una forma de extraer y representar los requerimientos de Awareness utilizando la nueva extensión. Después de ello presentaremos un ejemplo de aplicación sin entrar en el detalle de los procesos de generación de código y transformación de modelos. Finalizaremos con las conclusiones de este capítulo.

2. UsiXML

UsiXML[1] [Limbourg et al., 2005], que significa **US***er* **I***nterface e***X***tensible* **M***arkup* **L***anguage* (Lenguaje de marcado extensible de definición de interfaces de usuario) es un Lenguaje de Representación de Interfaces de Usuario (UIDL) [Guerrero García et al., 2009] basado en XML.

El propósito de UsiXML es describir interfaces de usuario (UI) para múltiples contextos de uso, como pueden ser interfaces gráficas, interfaces auditivas, interfaces de caracteres o interfaces multimodales.

En otras palabras, las aplicaciones interactivas con diferentes tipos de técnicas de interacción, modalidades de uso y plataformas de cómputo pueden ser descritas de forma que preserven el diseño, independientemente de las características particulares de la plataforma física de cómputo.

UsiXML soporta el desarrollo dirigido por modelos al representar modelos de especificación como un enorme gráfo, la cual permite expresar las UI a múltiples niveles de abstracción, y las etapas del desarrollo a través del uso de reglas de reescritura condicionales sobre el propio gráfo que representa el sistema y su desarrollo.

Algunas de las ventajas del enfoque proporcionado por UsiXML son:

[1] `www.usixml.org`

- El compromiso con la ontología: el lenguaje puede juzgarse desde todas las dimensiones de su definición, desde los conceptos hasta la sintaxis concreta, desde las tareas y el dominio hasta las UI concretas.

- Permite utilizar transformaciones entre modelos a diferentes niveles de abstraccion.

- Descompone las transformaciones en trozos manejables a nivel metodológico.

- Proporciona flexibilidad al permitir múltiples puntos de entrada y de salida [Limbourg et al., 2005].

- Facilita la interoperabilidad entre herramientas ya que incorpora modelos formalizados para favorecer el intercambio.

- Extendibilidad, UsiXML fue planeado para recibir diferentes contribuciones (3D, multi modalidad, interacción multi capa, etc.).

- Trazabilidad de las decisiones del diseño.

Actualmente la versión oficial y pública del lenguaje UsiXML es la 1.8[2]. Sin embargo, el lenguaje UsiXML está en un proceso de estandarización en coordinación con universidades y empresas para obtener una base sólida y flexible sobre la cual desarrollar aplicaciones interactivas multi contexto y multi modales.

Para entender el funcionamiento de UsiXML y de su metodología a nivel general, presentamos con un mayor detalle su arquitectura y su proceso de trabajo general (Sección 2.1). Posteriormente presentamos el metamodelo de UsiXML, el cual es la base descriptiva (Sección 2.2). Finalmente presentamos el proceso básico de transformación de modelos utilizado por UsiXML (Sección 2.3).

[2] `www.usixml.org`

2.1. Arquitectura y proceso general

UsiXML está basado en el marco de referencia Cameleon [Calvary et al., 2003], el cual define los pasos para desarrollar UI para aplicaciones interactivas multicontexto.

Cameleon identifica 4 medelos para definir un sistema:

1. Las Tareas y Conceptos (TyC), que representa las tareas que son llevadas a cabo por los usuarios y los conceptos del dominio requeridos por las tareas realizadas.

2. La interfaz de usuario abstracta (AUI), que define espacios de interacción que son independientes de la modalidad y de la plataforma combinando varias tareas de acuerdo a ciertos criterios. Esto se consigue a través de los Objetos Abstractos de Interacción.

3. La interfaz de usuario concreta (CUI), que representa una AUI a niveles inferiores de abstacción, para un contexto de uso específico, para una modalidad concreta e independiente de la plataforma, y a través de Objetos de Interacción Concretos.

4. La interfaz de usuario final (FUI), la cual representa la UI operativa con la cual interactuarán los usuarios.

Para la definición de estas etapas y su tratamiento en conjunto, UsiXML proporciona varios modelos para tratar con las UI multimodales y multiplataforma, así como un sistema de transformación para crear modelos intermedios, enlaces tipo *mappings* [Montero et al., 2006] y código.

La arquitectura general de Cameleon se muestra en la Figura 68. Una vez definidas las tareas y el dominio se realizan varias transformaciones para obtener la AUI. A partir de ahí se continua con la CUI para finalmente obtener la FUI.

Para ir de una etapa a otra es necesario transformar los modelos previos, ya

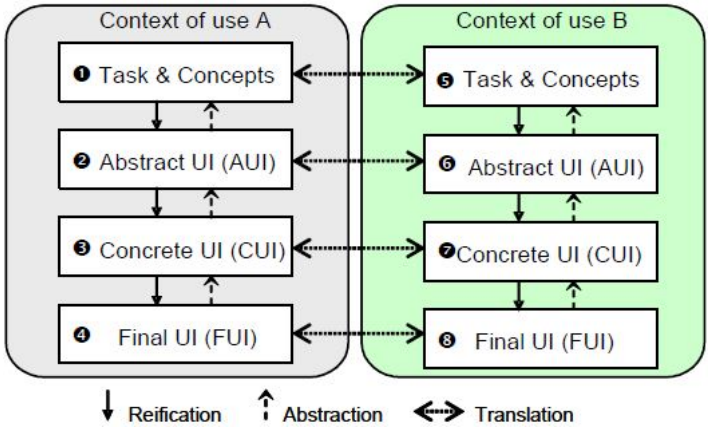


Figura 68. Marco de Referencia Cameleon.

sea de forma automática o manual. El proceso se realiza a través de la creación de enlaces o *mappings* entre los diferentes modelos, a fin de crear una estructura de seguimiento y enlazado que va desde las tareas y conceptos hasta los elementos de las interfaces de usuario finales.

En medio de todo el proceso interviene la definición del contexto, los recursos disponibles y cualquier otro modelo adicional que se haya agregado como parte de la base conceptual.

UsiXML adopta prácticamente todos los conceptos y principios de Cameleon y los concentra en un lenguaje de representación de interfaces de usuario basado en XML. Las herramientas disponibles proporcionan mecanismos de transformación orientados a distintos propósitos y plataformas, como son la Web o el Escritorio.

2.2. Metamodelo

Según describe la documentación oficial[3], ocho modelos distintos conforman el lenguaje UsiXML. Éstos modelos, mostrados en la Figura 69 son los siguientes:

- domainModel, el cual describe las clases de objetos manipulados por el usuario mientras interactua con el sistema.

[3] `http://www.usixml.org/index.php?mod=download&file=usixml-doc/UsiXML_v1.8.0-Documentation.pdf`

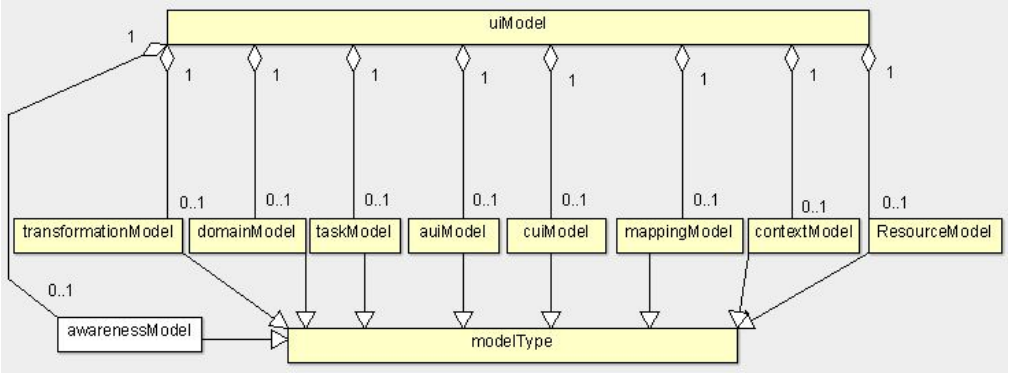


Figura 69. Metamodelo de UsiXML mostrando también el Modelo del Awareness.

- taskModel, el cual permite describir las tareas interactivas tal y como son vistas por el usuario que interactua con el sistema.

- auiModel, el cual representa la expresión canónica de de las interpretaciones (*renderings*) y manipulaciones de los conceptos y funciones del dominio de una manera que es independiente a cualquier modalidad de interacción y plataforma de cómputo.

- cuiModel, el cual representa la concretización de la interfaz de usuario abstracta. La CUI incluye la modalidad de la interfaz de usuario, como puede ser gráfica, auditiva o de caracterés. Es la representación más próxima de las interfaces de usuario finales.

- mappingModel, el cual contiene una serie de relaciones entre distintos modelos o elementos del modelado que están semánticamente relacionados.

- contextModel, el cual describe varios aspectos del contexto de uso en el cual un usuario ejecuta una tarea interactica sobre una plataforma y entorno específico. Contiene el modelo del usuario, el modelo de la plataforma y el modelo del entorno.

- ResourceModel, el cual contiene definiciones de recursos anexados a objetos de interacción abstractos o concretos los cuales son dependientes de aspectos de localización, como puede ser el lenguaje, la cultura, la forma

de lectura, etc.

- transformationModel, el cual contiene un conjunto de reglas que permite transformar una especificación (a cierto nivel de abstracción) a otra, o adaptar una especificación a un contexto de uso.

El enfoque de desarrollo de UsiXML se basa en describir dominio, tareas interactivas, contexto de uso y transformaciones sobre los mismos para obtener la aplicación deseada.

A continuación describimos de manera general cada uno de los modelos de UsiXML mencionados anteriormente.

2.2.1. *Modelo del dominio*

El modelo del dominio permite describir la estructura de las clases que conforman el dominio del sistema y que están relacionadas entre sí de forma similar al diagrama de clases UML, en el cual está basado. Además, el modelo del dominio permite describir los objetos o instancias de las clases del dominio.

Al ser UsiXML un lenguaje bien definido y delimitado por un esquema XML, la estructura de clases del dominio, los objetos o instancias que lo componen y las relaciones entre ellos se manejan con definiciones creadas en XML.

Este modelo forma parte de la primera etapa (Tareas y Conceptos) del marco de referencia Cameleon y en general, su definición forma parte de las primeras etapas de cualquier metodología de desarrollo.

Ya que UsiXML está hecho para definir interfaces de usuario, incluye también el modelado del dominio como base estructural de los datos manejados por la UI, los cuales son los que los usuarios utilizarán durante todo el sistema.

Otros marcos conceptuales para el modelado de sistemas incluyen aspectos más específicos como la sesión y la estructura jerárquica de los usuarios [Gutiérrez et al., 2006]. UsiXML carece de varios de estos modelos, como por

ejemplo, el modelo de sesión. Sin embargo, dada la flexibilidad del lenguaje, estos modelos pueden añadirse al lenguaje base o emularse a través del modelo del dominio. En cualquier caso, se necesitarán nuevas reglas de transformación para tener los nuevos modelos en cuenta.

2.2.2. *Modelo de tareas*

El modelo de tareas de UsiXML está basado en la notación ConcurTaskTree (CTT) [Paternò, 2004], utilizada para describir la estructura de tareas de un sistema interactivo. En el Capítulo 2 Sección 4.3 se describe la nomenclatura CTT.

En UsiXML, el modelado de las tareas es una parte fundamental de la metodología, ya que de la estructura jerárquica de las tareas puede obtenerse gran parte de la interfaz de usuario abstracta. La definición de la estructura de tareas en UsiXML puede hacerse a mano o utilizando herramientas gráficas parecidas a CTTE[Paternò et al., 2001] para facilitar esta labor.

Una tarea se aplica en un contexto de uso según lo descrito en la Sección 2.2.5. Además, está relacionada con la interfaz de usuario abstracta y con elementos del dominio.

2.2.3. *Modelo de la interfaz de usuario abstracta*

La intefaz de usuario abstracta (AUI) define espacios de interacción o unidades de presentación al agrupar tareas de a cuerdo a ciertos criterios (por ejemplo, patrones estructurales del modelo de tareas, análisis de carga cognitiva, identificación de relaciones semánticas, relaciones temporales, etc.).

También define un esquema de navegación entre los espacios de interacción y Objetos Abstractos de Interacción para cada concepto, de tal forma que son independientes de cualquier modalidad.

Una AUI abstrae un conjunto de interfaces de usuario concretas (CUI) en una definición que es independiente de cualquier modalidad de interacción (por ejem-

plo, interacción gráfica, interacción por voz, interacción basada en video, realidad virtual, realidad aumentada, etc.). Una AUI también puede ser considerada como una expresión canónica de la representación de los conceptos del dominio y de las tareas de una forma que es independiente de cualquier modalidad de interacción.

Esta etapa es esencial cuando se trabaja con sistemas que requieren ser multimodales, ya sea por ampliar sus capacidades de interacción o por estar orientados a modalidades poco convencionales, como la interacción por voz.

2.2.4. *Modelo de la interfaz de usuario concreta*

La interfaz de usuario concreta (CUI) fija una interfaz de usuario abstracta para un contexto de uso a través de objetos concretos de interacción, además de definir la disposición de los *widgets* y la navegación de la interfaz.

La CUI abstrae una interfaz de usuario final (FUI) en una definición que es independiente de la plataforma de cómputo. Aunque una CUI hace explícita la interfaz de usuario, sigue siendo una especie de maqueta que se ejecuta solamente en un entorno determinado, ya que todavía no es la interfaz de usuario final que puede ejecutarse en la plataforma de cómputo objetivo.

2.2.5. *Modelo del contexto*

El modelo del contexto describe tres aspectos del contexto de uso en el cual el usuario final ejecuta una tarea interactiva sobre una plataforma específica en un entorno determinado.

La versión usada de UsiXML (versión 1.8) proporciona en el modelo de contexto la forma básica del modelo del usuario, pudiendo definir estereotipos de usuario.

Aspectos de las plataformas de uso también son representados en este modelo, elementos tales como la plataforma software, hardware, el entorno de red, entre otros.

Finalmente se incluye una representación básica del entorno físico en el cual puede encontrarse el usuario.

El modelo de contexto, entonces, está formado por el modelo de usuario, modelo de la plataforma y el modelo del entorno.

Este modelo permite orientar un conjunto de tareas a un contexto específico. De esta manera, las interfaces de usuario concretas puede orientarse a distinas plataformas, usuarios o entornos.

2.2.6. *Modelo de mapeo*

El modelo de mapeo (*mappingModel*) contiene una serie de enlaces o *mappings* entre modelos o elementos de modelos.

Básicamente se utiliza para ligar elementos del modelado a lo largo de la metodología, permitiendo la trazabilidad entre las interfaces de usuario finales y las primeras etapas de la metodología, como son las tareas y los conceptos del dominio.

También le da flexibilidad y extensibilidad al propio UsiXML, de tal forma que otros modelos pueden agregarse al lenguaje base y ligarse a los modelos por defecto a través de nuevos enlaces definidos en este modelo.

Las relaciones de abstracción y reificación, así como la observación de los conceptos del dominio se definen a través de los elementos de este modelo.

2.2.7. *Modelo de recursos*

El modelo de recursos se incluye en UsiXML para poder incluir recursos que dependenden de la localización. El uso más notorio es el de internacionalizar interfaces de usuario.

2.2.8. *Modelo de transformación*

El modelo de transformación sustenta las reglas para transformar modelos de un nivel de abstracción a otro o para adaptar el sistema a los distintos contextos de uso definidos previamente.

La ventaja de este enfoque es que los mecanismos de transformación pueden anexarse al sistema tal como la definición del dominio y de las tareas. De esta forma, si otros modelos son anexados al lenguaje base, los mecanismos de transformación pueden actualizarse y utilizarse para generar una versión mejorada del sistema que ya está en uso.

2.3. Procesos de transformacion

El lenguaje UsiXML ha sido diseñado con una estructura subyacente de grafo. Por consiguiente, cualquier regla de transformación de grafos puede ser aplicada a la especificación de UsiXML.

UsiXML soporta un sistema de transformación compuesto por reglas de reescritura condicional del grafo que representa el lenguaje.

Este sistema de transformación está compuesto por varias reglas de transformación. Una regla es una regla de reescritura del grafo equipada con una condición negativa de aplicación y condiciones de atributos.

La Figura 70 muestra cómo un sistema de transformación es aplicado a una especificación UsiXML:

Sea G una especificación UsiXML (representada como un grafo), cuando **1)** una rama "Left Hand Side" (LHS) coincide en G y **2)** una condición "Negative Application Condition" (NAC) no coincide en G (varias NAC pueden estar asociadas a una sola regla) **3)** el LHS es reemplazado con una rama "Right Hand Side" (RHS). G entonces es transformado en G, una especificación UsiXML resultante. Todos los elementos de G no cubiertos por la coincidencia quedan intactos (sin cambios). Todos los elementos contenidos en la coincidencia LHS y no con-

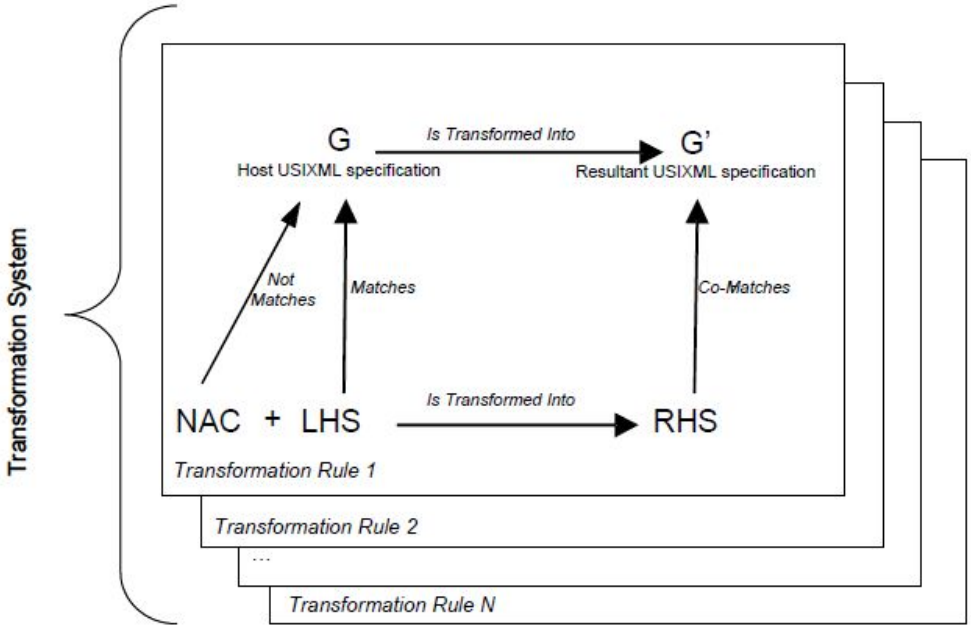


Figura 70. Sistema de transformación usado por UsiXML.

tenidos en la RHS son considerados como eliminados (las reglas tienen poderes destructivos). Para agregar más poder expresivo a las reglas de transformación, los atributos en una LHS pueden tener variables asociadas. Estas variables son inicializadas en la LHS y su valor puede ser usado para asignar un atributo dentro de la expresión del RHS. Una expresión también puede definirse para comparar una variable declarada en una LHS con una constante o con otra variable. Este mecanismo es llamado "condición de atributo".

Un desarrollador puede aplicar una o varias transformaciones a la especificacificación en UsiXML, ya sea para abstraerla o reificarla, o para adaptarla a nuevos contextos de uso.

A continuación describimos la extensión al lenguaje base de UsiXML a fin de modelar el Awareness como requerimiento de información en la primera etapa del marco de referencia Cameleon.

3. Extensión para el Soporte del Awareness

El soporte del Awareness se refleja tradicionalmente, y así lo recoge UsiXML, en las interfaces de usuario finales. Sin embargo, para poder llegar a ese punto, prácticamente todas las fases del desarrollo deben haber tocado el soporte del Awareness en algún punto.

La extensión descrita a continuación aumenta las capacidades de representación de UsiXML agregando un modelo especial para el Awareness relacionado con la primera etapa del marco de referencia Cameleon (Etapa de Tareas y Conceptos). Además, se extienden algunos modelos base, como es el modelo de Mapeo y el modelo de Contexto para incluir algunas entidades requeridas para el soporte base del Awareness.

El enfoque tomado para desarrollar esta extensión es el siguiente: el Awareness es un requerimiento de información relacionado con las necesidades de los usuarios durante el proceso de toma de decisiones dentro de una tarea. De esta manera, el Awareness está relacionado primeramente con los usuarios y sus objetivos y después con las tareas. Sin embargo, dado que ya existen transformaciones en UsiXML para propagar la observación de atributos de las clases del dominio, preferimos aprovechar el enfoque tomado por dichas transformaciones y proponer mecanismos que permitan aplicar transformaciones similares para la propagación de los requerimientos de Awareness..

La versión utilizada del lenguaje UsiXML es la 1.8, que en el periodo de este trabajo de tesis es la versión pública oficial.

Siguiendo el diseño de UsiXML, hemos agregado un nuevo modelo al metamodelo base de UsiXML (Figura 71). El modelo del Awareness, representado por la entidad *awarenessModel*, el cual, como todos los demás modelos base de UsiXML desciende del *modelType* y una definición en UsiXML contiene cero o un modelo del Awareness.

El Modelo del Awareness puede agregar cero o varios Requerimientos de Awa-

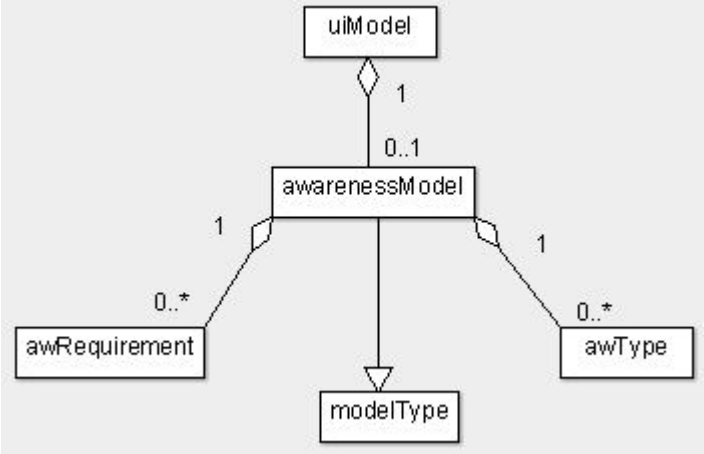


Figura 71. Modelo del Awareness en UsiXML.

reness (*awRequirement*) y cero o varios Tipos de Awareness *awConcreteType*), según lo explicado en el Capítulo 4 sobre estos términos.

A continuación describiremos primeramente los modelos base de UsiXML que fueron extendidos con alguna entidad, de tal manera que las entidades o relaciones añadidas tengan sentido dentro de ese modelo y no dentro del modelo del Awareness.

Posteriormente presentamos el modelo del Awareness y del requerimiento de Awareness, el cual utiliza varias de las entidades presentadas anteriormente.

3.1. Extensión a los modelos base

Varias entidades han sido agregadas a los modelos base de UsiXML debido a que son requeridas para el soporte del Awareness y a que conceptualmente no forman parte del modelo del Awareness.

La entidad *selectionConstraint* (Figura 72)ha sido añadida al modelo del dominio para representar restricciones de acceso a las instancias de una clase del dominio. Por ejemplo, la restricción *"edad mayor a 18 años"* puede ser creada para la entidad *Estudiante*. Esta entidad representa la primera forma de manejar la privacidad de datos en lo que respecta al Awarenes. Siguiendo la línea de

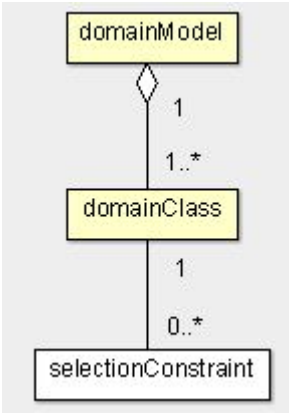


Figura 72. Entidad *selectionConstraint* agregada al modelo del dominio.

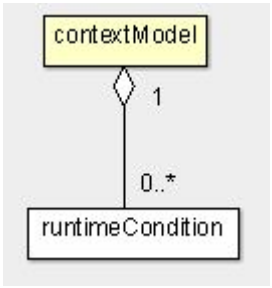


Figura 73. Entidad *runtimeCondition* agregada al modelo del contexto.

[Drury and Williams, 2002], los datos de Awareness sólo se proporcionan a quien lo requiere y sólo lo que requiera.

La entidad *runtimeCondition* (Figura 73) ha sido agregada al modelo del contexto para representar condiciones que puedan necesitar acceso a datos que sólo están disponibles en tiempo de ejecución. Como ejemplo de estos datos dinámicos encontramos el número de usuarios conectados, la cantidad de veces que se ha realizado una consulta, la hora actual, el nombre del usuario conectado, etc. Estas condiciones se caracterizan por acceder a varias fuentes de datos del sistema, del dominio o del entorno actual de ejecución para obtener un valor aprovechable en una operación lógica de la cual se obtiene un Verdadero o Falso, Sí o No, etc.

Esta entidad representa la primera forma de controlar la distracción y la

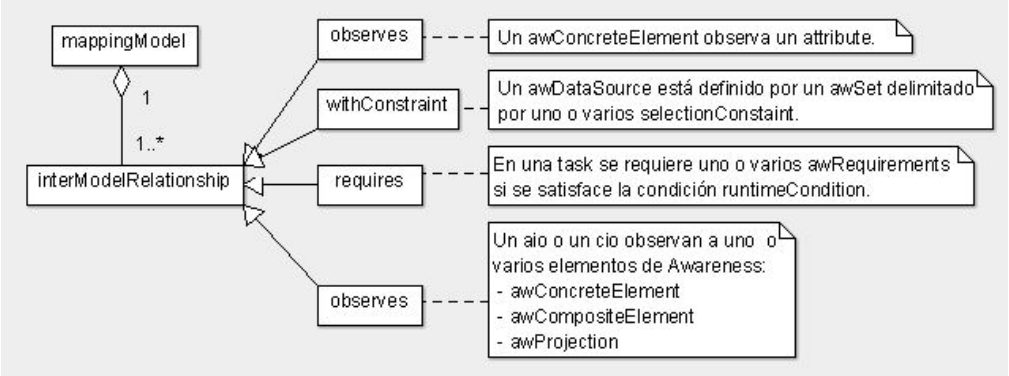


Figura 74. *Mappings* o relaciones entre elementos del modelado.

sobrecarga cognitiva al poder restringir cuándo se aplica un determinado requerimiento de Awareness y así controlar cuándo los usuarios reciben datos.

En la descripción del Requerimiento de Awareness puede entenderse mejor el papel de la *runtimeCondition* en el soporte del Awareness.

Varios *mappings* o relaciones entre modelos fueron agregados al modelo de mapeo o se extendieron sus usos. Los *mappings* que fueron cambiados o agregados (Figura 74) son los siguientes:

- *observes*: es utilizado por UsiXML para ligar elementos de la interfaz de usaurio abstracta o concreta a conceptos del dominio, como los atributos de una clase del dominio. Para el soporte del Awareness se utilizado para ligar elementos de Awareness con atributos de una clase (Figura 75a) y para ligar elementos de la interfaz de usuario abstracta o concreta con los elementos de Awareness (Figura 75), encadenando desde la interfaz de usuario final hasta los elementos de Awareness.

- *withConstraint*: utilizado para definir una fuente de datos de Awareness. Permite tomar un conjunto de elementos del mismo Tipo de Awarenes y ligarlos con una restricción de acceso a datos de una clase del dominio, creando así lo que se asemeja a una vista o clase derivada.

- *requires*: utilizado para definir que una tarea tiene un Requerimiento de

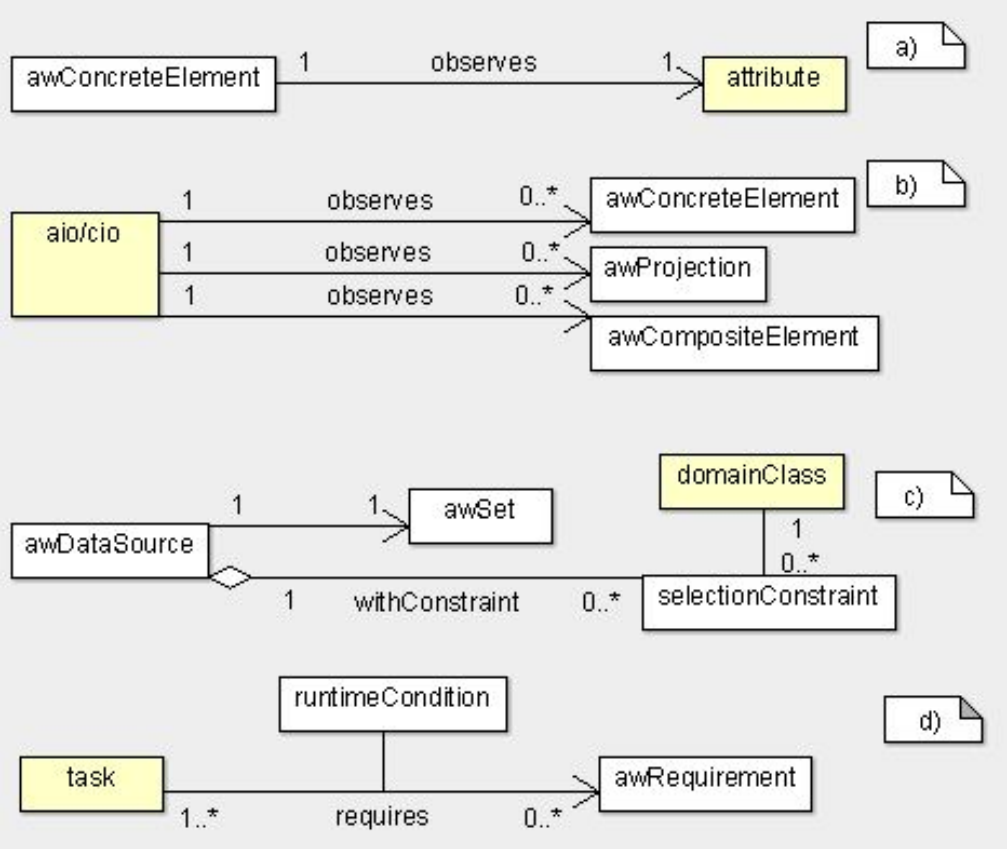


Figura 75. Relaciones entre distintos modelos a través de los *mappings* **a,b)** *observes*, **c)** *withConstraint* y **d)** *requires*.

Awareness en una determinada situación, la cual está definida por una restricción en tiempo de ejecución (*runtimeCondition*).

Recordemos que estos *mappings* cubren el soporte básico del Awareness. Nada impide agregar otros mappings que enlazen por ejemplo las interfaces de usuario concretas con entidades que definan atributos sobre la transmisión de los datos de Awareness como la frecuencia de actualización, disparadores para iniciar o detener la transmisión de datos, etc. Al pertenecer a la etapa de diseño, se incrementan bastante las posibles formas de consiguir los mismos resultados.

En cualquier caso, el Capítulo 3 Sección 6 ofrece una guía sobre varios aspectos importantes del Awareness directamente reflejados en las interfaces de usuario finales.

A continuación presentamos el modelo del Awareness, que es donde se definen los principales aspectos del Awareness como los Requerimientos de Awareness y los Tipos de Awareness.

3.2. Modelo del Awareness

La definición del Awareness a nivel conceptual es similar a la definición del dominio de trabajo. Se habla en términos de información requerida de cierta entidad, pero no se va más allá en cuestión de detalle.

Un Tipo de Awareness representa estado de conocimiento o Awareness que se tiene de los atributos de algún o algunos entes (objetos o instancias en el caso de un sistema informático).

Por ejemplo, cuando se va caminando, se tiene un "Awareness del camino", lo cual incluye el tipo de camino que es, si es recto o curvo, si tiene obstáculos, si es angosto, etc. En conjunto todos estos atributos se refieren a la misma entidad o tipo de entidad, por lo cual pueden englobarse en el "Awareness del camino".

Si estuvieramos trabajando en un dominio con robots que tienen que seguir un camino determinado, seguramente estos robots deberán de tener Awareness

del camino. Probablemente en la definición de su dominio existan las entidades para manejar la información del camino capturada por los sentidos del robot. El Awareness en este caso está más relacionado con la necesidad de información que con la definición del dominio con entidad en donde físicamente se mantiene la información del camino.

Está claro que si se requiere Awareness de algo, esa información debe estar en el sistema, ya sea en las entidades del dominio o en las entidades del sistema. Por ejemplo, la sesión normalmente es manejada como parte del dominio, aunque algunas arquitecturas definan la sesión como parte del sistema y no del dominio.

En UsiXML sólo existe el metamodelo base y las entidades que se definan en el dominio. Por lo tanto, serán las que proporcionen la información de Awareness requerida.

La definición de los Tipos de Awareness se realiza por separado y esto se debe a varias razones:

- El modelo de los Tipos de Awareness representa la información de Awareness requerida, sin más, la cual está relacionada con alguna(as) entidad conceptual y de interés durante una tarea. De forma distinta, las entidades del dominio y del sistema representan otra estructura de información más ligada a los datos de trabajo que a las necesidades de información, además está sujetas a otras restricciones arquitectónicas y de optimización como la normalización.

- Los Tipos de Awareness estarán directamente ligados a los Requerimientos de Awareness, por lo cual cualquier mecanismo creado para soportar el Awareness deberá quedar también ligado a los Requerimientos de Awareness y a los Tipos utilizados.

- La separación conceptual permite tratar cada cosa de forma adecuada, sin mezclar información para la toma de decisiones (Awareness) con la información de trabajo (dominio) y del sistema, aunque se entrelacen en

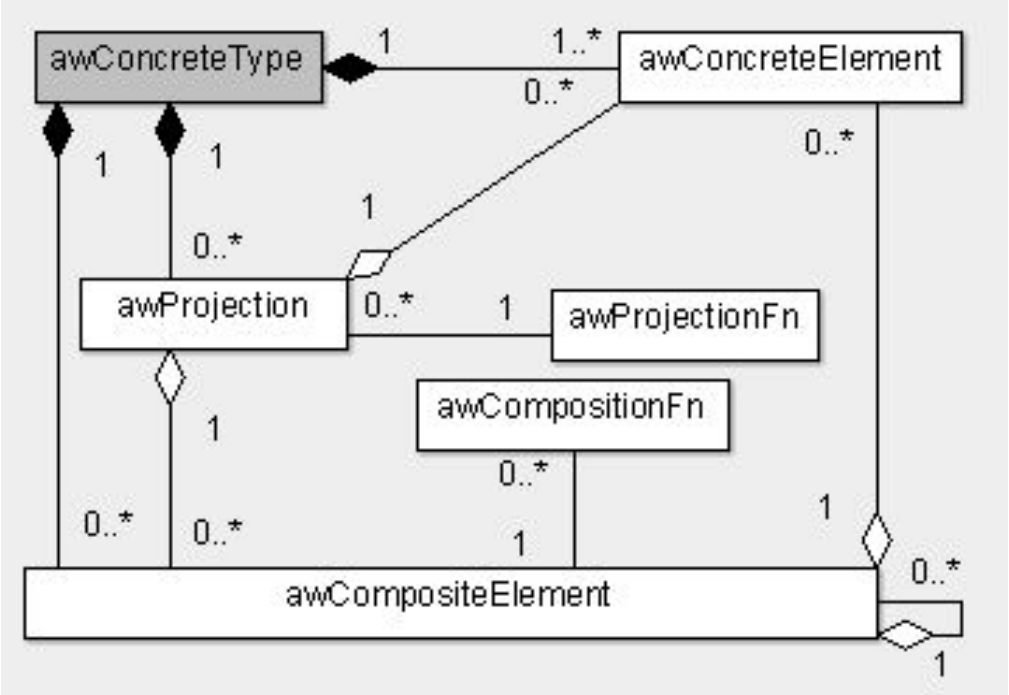


Figura 76. Modelo para la definición de los tipos de Awareness.

varias ocasiones.

Un Tipo de Awareness puede actuar como una clase del dominio de la cual se extraen datos contínuamente para dárselos a los usuarios que lo necesitan. La forma de darle soporte al Awareness puede variar mucho entre plataformas y contextos de uso. Lo que no varía es que la información de Awareness debe estar bien identificada y justificada para que el sistema no le de soporte a necesidades de información inexistentes o que se pierden en las diferentes etapas del desarrollo del software.

Para describir un Tipo de Awareness proporcionamos la entidad *awConcreteType* (Figura 76), la cual está compuesta por varios elementos de Awareness acorde con la teoría del Awareness de Situación [Endsley, 1995]. Los elementos de Awareness son:

- *Elementos concretos.* Representa atributos de la entidad observada, los

cuales pueden percibirse de forma directa. Por ejemplo, Anchura, Longitud, Altura, Peso, Color, Forma, Tono, Temperatura, etc. Representan atributos que son manipulados por el sistema. Entidad *awConcreteElement*.

- *Elementos compuestos*. Representan atributos cuyo valor es derivado o calculado del valor de otros atributos. Se consideran un nivel superior de abstracción, dado que su valor depende de otros. Como ejemplo de ellos, el elemento Velocidad, cuyo valor es calculado de la Distancia y del Tiempo, o el elemento NombreCompleto, cuyo valor se calcula del nombre y de los apellidos. La entidad *awCompositeElement* representa a los elementos compuestos, mientras que la entidad *awCompositionFn* representa la función de composición.

- *Proyecciones*. Representa una proyección de valores en un momento futuro. Una proyección debe retornar un conjunto de valores de distintos atributos proyectados. Por ejemplo, el estado del tiempo (clima) para un momento futuro proyecta los valores de la temperatura, humedad, probabilidad de lluvias, etc. La entidad *awProjection* representa la proyección y la entidad *awProjectionFn* representa el cálculo para proyectar los valores de los elementos concretos o compuestos que quieren proyectarse.

  Para soportar proyecciones de Awareness, puede ser necesario tener a disposición el historial de valores de los elementos de awareness que componen cada proyección, ya que varias funciones de proyección se basan en los valores de datos históricos.

A continuación presentamos el modelo de Requerimiento de Awareness.

3.3. Requerimiento de Awareness

Un Requerimiento de Awareness (AR) expresa un estado de conciencia o conocimiento requerido por actor durante el desarrollo de una tarea. Es decir, es un requerimiento de información actualizada durante una tarea.

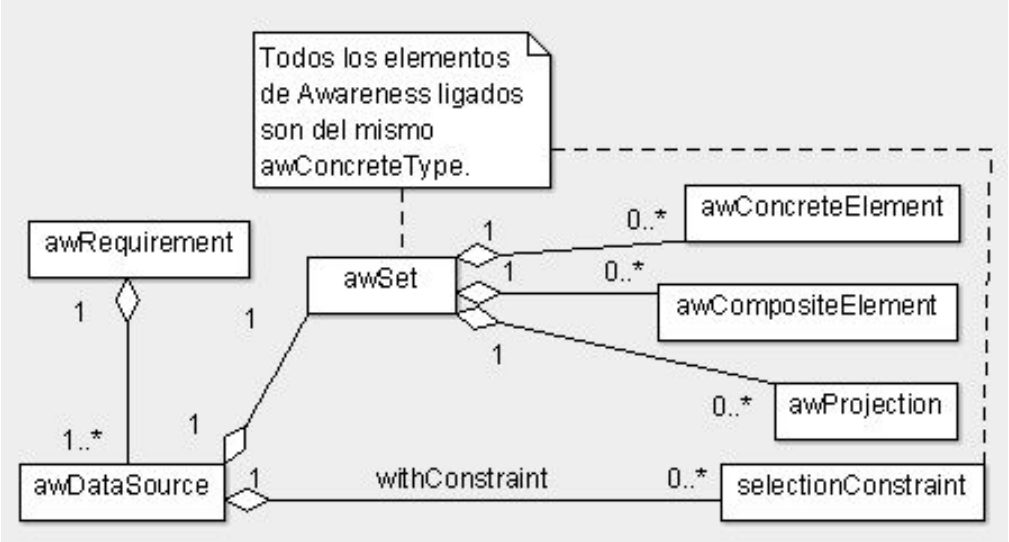


Figura 77. Modelo del Requerimiento de Awareness a través de las nuevas entidades integradas a UsiXML a través del modelo del Awareness.

Para representar un requerimiento de Awareness, primero puede realizarse una definición informal del mismo, la cual exprese en lenguaje natural o similiar la información que se requiere y todos los aspectos referentes a dicho requerimiento (Véase Capítulo 4 Sección 4).

A partir de esta definición se utilizan los siguientes modelos para definir las entidades intermedias que servirán para definir el requerimiento de Awareness (Figura 77):

- *awSet*: Agrupa elementos de un mismo Tipo de Awareness. Permite formar grupos de elementos del Awareness que deben transmitirse al usuario en un determinado requerimiento de Awareness.

- *awDataSource*: Crea una grupo de datos de Awareness usando un *awSet* y un *selectionConstraint* para restringir los datos a los que se pueden acceder. Un *awDataSource* define un conjunto de datos de Awareness estructurados y delimitados que es requerido por uno o varios usuarios.

- *awRequirement*: Representa un conjunto de fuentes de datos de Awareness

(*awDataSource*), los cuales serán proporcionados al usuario cuando éste lo requiera. Contesta a la pregunta de qué datos se le deben proporcionar a un usuario específico en una determinada tarea y situación.

3.4. Soporte general del Awareness

El soporte general del Awareness en UsiXML usando la extensión propuesta se expresa en los siguientes pasos:

1. Se identifican y documenta los requerimientos de Awareness de forma informal, como cualquier otro requerimiento. Puede utilizarse como guía el formato propuesto en el Capítulo 4 Sección 4.1 para representar requerimientos de Awareness.

2. Se definen los Tipos de Awareness en base a la información requerida.

3. Se caracterizan los Tipos de Awareness con las entidades del dominio ligando los elementos concretos del Awareness con atributos de las clases del dominio.

4. Se definen los grupos de elementos de Awareness que van a proporcionarse a la vez y que pertenecen al mismo Tipo de Awareness.

5. Utilizando los grupos de elementos, se definen las fuentes de datos de Awareness en base a las restricciones de acceso a los datos observados según lo documentado en el requerimiento de Awareness.

6. Se crean los requerimientos de Awareness a nivel de modelado usando las fuentes de datos de Awareness. Hasta este punto se tiene identificada, estructurada y delimitada la información de Awareness que se va a proveer a través de las interfaces de usuario.

7. Se enlazan los requerimientos de Awareness con las tareas en donde se requiere dicha información. El Awareness será propocionado según el con-

texto de uso y las condiciones en tiempo de ejecución. Además, estas restricciones permiten proporcionar Awareness sólo a quien lo necesita.

8. Se utilizan los mecanismos de transformación adaptados de UsiXML para generar las interfaces de usuario abstractas y concretas tomando en cuenta el Awareness de la misma forma que ya se enlazan los conceptos del dominio (atributos de clases) ligados a una tarea.

9. Se refina el diseño de las interfaces de usuario que proveen Awareness para manejar restricciones más específicas como el ritmo de observación y transmisión de datos al usuario, entre otras. (Ver Capítulo 3 Sección 6).

En la siguiente sección aplicaremos este proceso a un caso de estudio con el cual se explica más a detalle el uso y los modelos del Awareness.

4. Ejemplo

En esta sección explicaremos los modelos del Awareness utilizando un sistema sencillo en el cual están presentes varios requerimientos de Awareness y todos los elementos del Awareness.

El sistema utilizado para el ejemplo busca apoyar las reuniones de astrónomos aficionados que viven en distintas ciudades. Un sistema tipo chat mejorado con algunas características que cumple las necesidades del grupo, las cuales son:

- Tener un punto de reunión para que los miembros del grupo de aficionados puedan comunicarse de forma escrita y así poder planificar observaciones en la localización de alguno de ellos.

- Tener disponible la información meteorológica correspondiente a los nublados en la localización de cada usuario conectado y dispuesto a recibir compañeros.

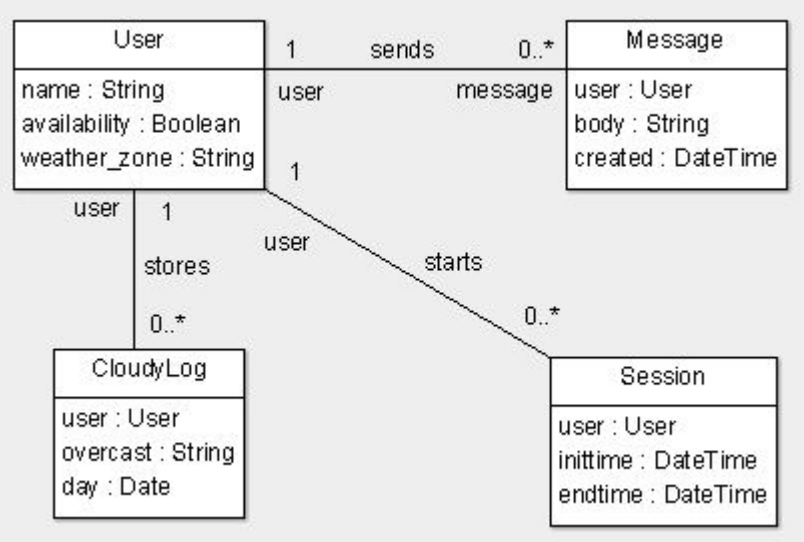


Figura 78. Dominio del sistema de apoyo al grupo de astronomos aficionados.

- Tener disponible bajo demanda la información de nublados en el futuro, es decir, el pronóstico de los nublados en cierto rango de tiempo. Es importante que el sistema no dependa enteramente de servicios externos que pueden dar esos pronósticos, sino también que provea un pronóstico propio en base a los datos históricos en el sistema.

4.1. Tareas y dominio

Viendo las necesidades anteriores podemos definir las partes básicas para un sistema de este tipo, cuya tarea principal llamaremos `chat` y en la cual se desarrolla la comunicación entre los usuarios y la cual puede llamar a la tarea `userforecast` para conocer a detalle el estado futuro aproximado de los nublados en la localidad de un usuario específico.

Detectamos cuatro entidades que conforman el dominio para el sistema de ejemplo. El diagrama del dominio se muestra en la Figura 78 y las clases se detallan en las Tablas 30, 31, 32 y 33.

Para los propósitos del ejemplo nos centraremos en el Awareness relacionado con la información de los usuarios y la información de los pronósticos del tiempo.

Clase *User*		
Atributo	**Tipo**	**Descripción**
name	String	Nombre del usuario.
availability	Boolean	Disponibilidad para recibir a otros compañeros en su localización para una observación astronómica en grupo.
weather_zone	String	Identificador de la zona geográfica en donde se encuentra el usuario a fin de poder consultar servicios externos conocer las condiciones meteorolgicas, especialmente los nublados.

Tabla 30. Maneja los datos de los usuarios del sistema.

Clase *Session*		
Atributo	**Tipo**	**Descripción**
user	User	Usuario al que pertenece la sesión. Sólo puede haber una sessión abierta para cada usuario.
inittime	DateTime	Momento de inicio de sesión
endtime	DateTime	Fin de la sesión. Se cierra cuando hay inactividad por mucho tiempo o cuando el usuario sale del sistema.

Tabla 31. Mantiene la información de las sesiones. De esta manera se puede conocer los usuarios que están en línea y los que están desconectados.

Otro Awareness importante es el de los mensajes, ya que a través de ellos es como los usuarios pueden comunicarse o enterarse del rumbo de la conversación.

Las tareas principales del sistema son:

- `chat`: Es la tarea en donde se desarrolla la comunicación tipo chat público parecido al IRC (mensajes públicos que llegan a todos los miembros conectados). Contiene varios requerimientos de Awareness, pero nos centraremos en los que tienen que ver con la información de los usuarios.

- `userforecast`: Tarea informativa que muestra información sobre el estado del tiempo en un futuro cercano. Servirá para explicar las proyecciones de Awareness. Sólo tiene sentido para los usuarios que estén disponibles para recidir personas, y por lo tanto, sólo ellos tendrán el requerimiento de Awareness proporcionado por esta tarea.

Clase *CloudyLog*		
Atributo	**Tipo**	**Descripción**
user	User	Usuario al que pertenece esta bitácora.
overcast	String	El estado del clima en formato texto para la fecha indicada.
day	Date	Fecha de la bitácora.

Tabla 32. Usada para guardar el estado del clima en la localización de un usuario. Sirve para hacer cálculos locales sobre el estado futuro del tiempo en la localización del usuario cuando no se tiene acceso a servicios externos de pronóstico del clima.

Clase *Message*		
Atributo	**Tipo**	**Descripción**
user	User	Creador del mensaje público.
body	String	Texto del mensaje.
created	DateTime	Fecha y hora de creación del mensaje.

Tabla 33. Usada para guardar los mensajes públicos que utilizan los usuarios para planear una reunión de observación en alguna localización de uno de los usuarios participantes y disponibles.

4.2. Análisis de requerimientos de Awareness

Para definir los requerimientos de Awareness en cada tarea utilizaremos el formato descrito en el Capítulo 4 Sección 4.1. Los requerimientos de Awareness encontrados se describen en las Tablas 34, 35 y 36.

Una vez identificados y analizados los requerimentos de Awareness, podemos empezar a definir los tipos de Awareness, los cuales representan la información requerida por los usuarios pero agrupada en base a la entidad conceptual a la cual pertenecen los atributos observados, ya sean idénticos a los de la entidad o entidades del dominio observadas o a niveles de abstracción mayor, como los elementos compuestos del Awareness.

4.3. Definición de los tipos de Awareness

Requerimiento de Awareness	
Id	*userReq*
Actores	Cada usuario del sistema.
Tarea	`chat`
Información requerida	De los usuarios, su nombre, disponiblidad para reuniones y el estado del tiempo (clima) en su zona.
Restricciones de datos	Sólo de los usuarios en línea y disponibles para recibir personas en su localidad.
Aplicabilidad	Durante toda la tarea.
Justification	El nombre de usuario es necesario para identificarlos, la disponibilidad para saber rápidamente quien está participando con ánimos de recibir personas, y el estado del tiempo ese día es para darse una idea de como está el clima en la zona de ese usuario.
Otros	Del clima sólo importan las nubosidades. Es decir, saber si está el cielo despejado o nublado.

Tabla 34. Requerimiento de Awareness de los usuarios participantes en la programación de observaciones astronómicas.

Requerimiento de Awareness	
Id	*cloudyExtReq*
Actores	Sólo los usuarios disponibles para recibir personas.
Tarea	`userforecast`
Información requerida	La proyección del estado del tiempo obtenida externamente.
Restricciones de datos	Sólo los datos de la zona del usuario del cual interesa conocer el clima de su zona.
Aplicabilidad	Durante toda la tarea, la cual en sí misma debe ser bajo demanda. Proporcionar este requerimiento si hay disponibilidad para obtener los pronósticos externamente, usando servicios web para ello.
Justificación	Las reuniones de observación de personas que viven en diferentes ciudades pueden planificarse para días posteriores. Por eso es importante conocer el pronóstico del clima para días futuros.
Otros	La modalidad de este sistema es gráfica. Utilizar ventanas externas para no interrumpir el flujo del sistema. Es decir, para no interrumpir el flujo de mensajes como información principal.

Tabla 35. Requerimiento de Awareness del estado del clima en alguna zona de interés ligada a un usuario. Necesaria para planificar observaciones.

Requerimiento de Awareness	
Id	*cloudyLocalReq*
Actores	Sólo los usuarios disponibles para recibir personas.
Tarea	`userforecast`
Información requerida	La proyección del estado del tiempo obtenida mediante los datos históricos almacenados en el sistema en la clase *CloudyLog*. Usar algoritmos de estimación del clima implementados en el propio sistema.
Restricciones de datos	Sólo los datos de la zona del usuario del cual interesa conocer el clima de su zona.
Aplicabilidad	Durante toda la tarea, la cual en sí misma debe ser bajo demanda. Proporcionar este requerimiento cuando no haya posibilidad de usar servicios de pronóstico externos.
Justificación	Las reuniones de observación de personas que viven en diferentes ciudades pueden planificarse para días posteriores. Por eso es importante conocer el pronóstico del clima para días futuros.
Otros	Debe informarse que el pronóstico del tiempo es local y no externo.

Tabla 36. Requerimiento de Awareness del estado del clima calculado localmente para proveer proyecciones del clima.

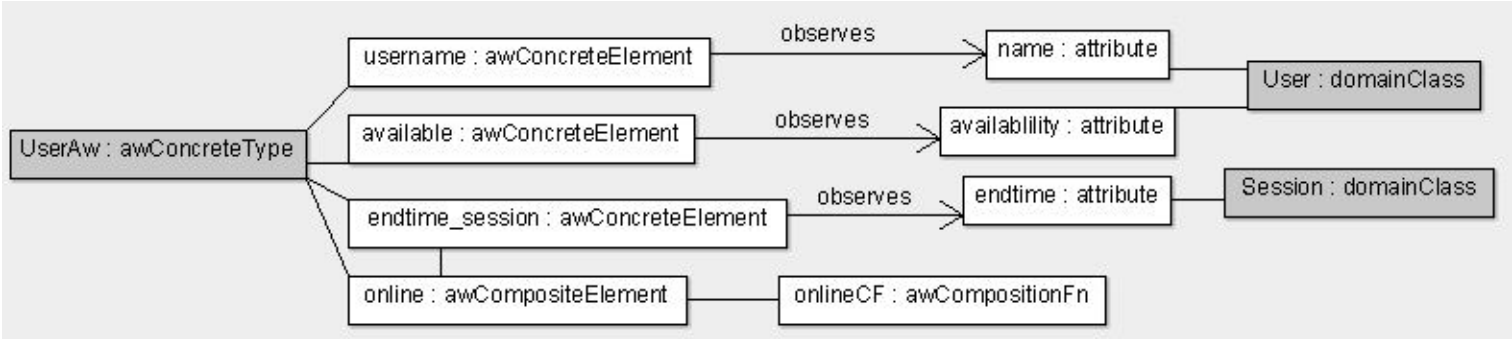


Figura 79. Tipo de Awareness *UserAw*, definido para representar el Awareness de los usuarios y caracterizado/enlazado con las entidades del dominio a través de los *mappings* "observes".

Para definir un tipo de Awareness comenzamos por identificar los elementos de Awareness que requiere el usuario de la entidad conceptual. Del usuario se requiere conocer su nombre, su disponibilidad para reuniones y si están en línea o no. También se requiere saber el estado de las nubosidades en su zona, sin embargo, esta característica pertenece más bien a la zona del usuario, no al usuario en sí, por lo que esta información pertenece a otro tipo de Awareness.

En la Figura 79 se muestra el Awareness de Usuario (*UserAw*) caracterizado con entidades del dominio de trabajo. Puede definirse primero el tipo de Awareness y después caracterizarse, pero por cuestiones de practicidad lo hacemos al mismo tiempo.

El *UserAw* está relacionado con las clases *User* y *Session* a través del *mapping "observes"*, lo cual significa que el *UserAw* agrupa elementos Awareness relacionados con esas dos entidades del dominio y que para efectos prácticos los datos de Awareness proveendrán de dichas clases.

Dos elementos concretos de Awareness, *username* y *available* provienen de la clase *User*. Después encontramos el elemento *endtime_session*, el cual proviene de la clase *Session*. Dichas clases del dominio están relacionadas de tal forma que un usuario puede tener varias sesiones pero sólo una abierta, por lo cual se puede utilizar el atributo *endtime* ligado al elemento *endtime_session* para saber si el usuario está en línea o no.

Ya que el elemento *endtime_session* no nos interesa, ¿para qué lo definimos como parte del User Awareness? Porque es necesario para calcular el elemento compuesto *online* usando la función de composición *onlineCF*. Dicha función toma el valor del elemento concreto *endtime_session* y devuelve un valor lógico (Verdadero o Falso), el cual es el que le interesa a los usuarios.

No podemos definir directamente un elemento concreto *online*, ya que no tenemos de donde obtener directamente su valor. Por el contrario, ya que se necesita un proceso de cálculo o inferencia para generar nuevo conocimiento,

requerimos primero tener a disposición los valores que se necesitan para calcular el nuevo valor. Es decir, todo elemento compuesto de Awareness requiere tener a disposición los valores de los elementos de Awareness de los cuales depende, ya sean elementos concretos o también compuestos. Por esta razón se incluye el elemento *endtime_session* al *UserAw*.

Esto no quiere decir que los valores del elemento *endtime_session* vayan a ser transmitidos al usuario. Esto se define mediante los Awareness Sets que explicaremos más adelante.

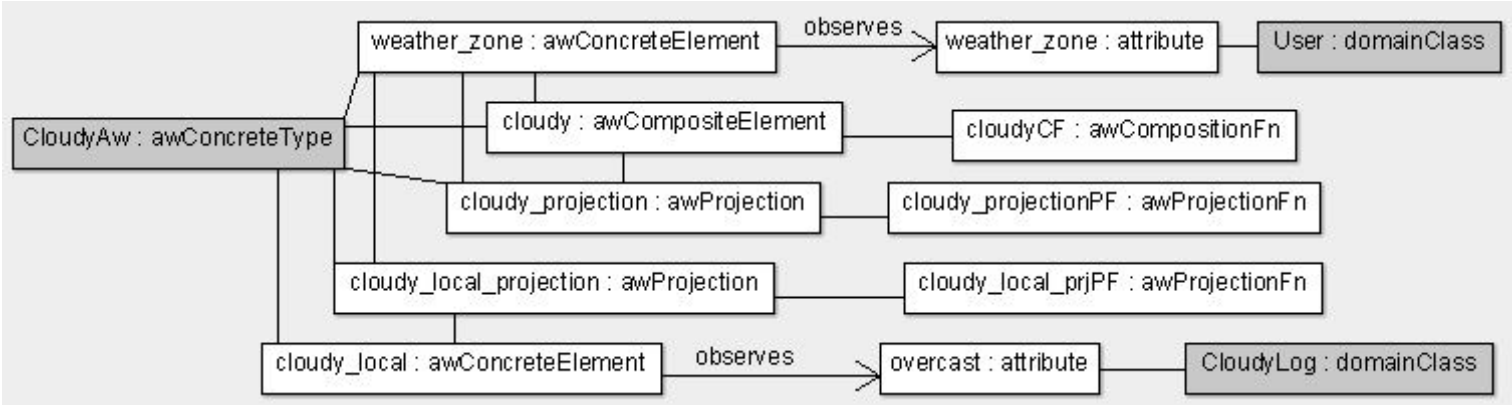


Figura 80. Tipo de Awareness *CloudyAw*, definido para representar el Awareness de las condiciones meteorológicas (los nublados). Nótese que mayormente define proyecciones, ya que es el Awareness requerido por los ususuarios.

El tipo de Awareness *CloudyAw*, mostrado en la Figura 80, representa la información referente a los nublados de una zona geográfica. Estas zonas están ligadas a los usuarios, pero además hay un historial de datos climatológicos a disposición para realizar predicciones si así se requiere.

Nótese que se utilizan nombres distintos para diferenciar a los elementos concretos de Awareness de los atributos de clase observados. Esto es para no confundir unos con otros. Sin embargo, al sistema no le afecta que se llamen igual, ya que son diferentes cosas.

El elemento concreto *weather_zone* observa los valores del atributo *weather_zone*. Sin embargo, su utilidad real es la de poder calcular el elemento compuesto *cloudy*, el cual representa las nubosidades en dicha zona. El proceso de cálculo lo realiza la función *cloudyCF*, la cual puede perfectamente consultar servicios externos para obtener su valor.

El elemento concreto *cloudy_local* observa los valores del atributo *overcast* de la clase *CloudyLog*, la cual también está relacionada con la clase *User*.

Los elementos restantes de Awareness en el *CloudyAw* son dos proyecciones. La proyección *cloudy_projection* será utilizada para proporcionar el valor de los elementos *weather_zone* y *cloudy* para varios días futuros, como si de un pronóstico del tiempo se tratara. En este caso, esta proyección utilizará servicios externos para calcular dichos valores. La entidad *cloudy_projectionPF* se representa este proceso.

La proyección *cloudy_local_projection* será utilizada para proporcionar el valor de los elemntos *weather_zone* y *cloudy_local* en varios días futuros, siendo la entidad *cloudy_local_prPF* la que representa este proceso de cálculo de los valores proyectados.

Ya que tenemos definidos y caracterizados los tipos de Awareness, procedemos a agrupar los elementos de Awareness que van a ser transmitidos al usuario al mismo tiempo.

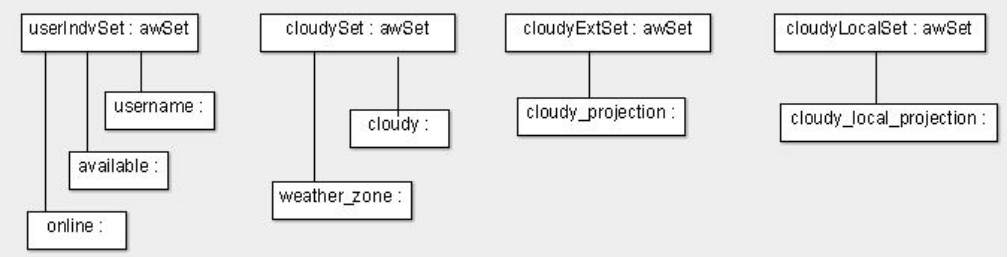


Figura 81. Awareness sets, definidos para representar grupos de elementos de Awareness, que son los que posteriormente se ligarán mediante *mappings* "observes" a objetos abstractos de interacción de la AUI.

4.4. Definición de los requerimientos de Awareness

El propósito de los Awareness Sets es agrupar elementos de Awareness que van a transmitirse al mismo tiempo y que pertenecen a al mismo tipo de Awareness.

Los Awareness Sets para nuestro ejemplo se muestran en la Figura 81 y se describen a continuación:

- *userIndvSet*: Agrupa los elementos del *UserAw* que serán proporcionados a los usuarios durante la tarea `chat`.
- *cloudySet*: Agrupa los elementos del *CloudyAw* que serán proporcionados a los usuarios durante la tarea `chat` a la vez que los del set *userIndvSet*.
- *cloudyExtSet*: Incluye la proyección que será proporcionada a los usuarios cuando lo pidan a través de la tarea `userforecast` cuando estén disponibles los servicios externos para el calcular los pronósticos climáticos.
- *cloudyLocalSet*: Incluye la proyección que será proporcionada a los usuarios cuando lo pidan a través de la tarea `userforecast` cuando **no** estén disponibles los servicios externos para el calcular los pronósticos climáticos y se tenga que utilizar el proceso de proyección interno del sistema.

La necesidad de definir Awareness Sets se deriva de que puede haber distintas fuentes de datos de Awareness para los mismos Sets. Además, permite definir

primero las estructuras de elementos que van a proporcionarse para luego poder crear una o varias fuentes de datos de Awareness.

UsiXML hace uso exhaustivo del *mapping* "observes" para relacionar atributos del dominio con las interfaces de usuario abstractas y concretas. De esta forma, una etiqueta puede "observar" el valor del atributo de una clase a través de este enlace, el cual permite trazar cada lugar en donde se observa dicho atributo y de forma inversa, qué atributos observa cada elemento de la interfaz. Este enlazado puede realizarse manualmente (no muy práctico) o de forma automatizada por las herramientas de desarrollo de UsiXML, que son las que aprovechan mejor este lenguaje.

Este mecanismo de enlazado trazable a través del *mapping* "observes" puede utilizarse igualmente para enlazar elementos del Awareness con las interfaces de usuario. A nivel de diseño, cada interfaz tendrá que manejar parámetros propios como la frecuencia de actualización del valor de la interfaz. Por ejemplo, no es lo mismo una etiqueta que transmite un contador, a una voz que transmite un contador.

A continuación definiremos las fuentes de datos de Awareness, las cuales están representadas por la entidad *awDataSource*.

Las fuentes de datos de Awareness (*awDataSource*) representan una conjunto de elementos de Awarenses del mismo tipo aunada a un conjunto de restricciones de selección de instancias de datos del dominio. Es decir, *awSet*s junto con predicados para seleccionar solamente determinadas instancias de la clases que alimentan los tipos de Awareness ligados a los elementos del Set.

Esta combinación permite crear una fuente de datos de Awareness la cual está conformada por varias instancias de los Awareness Sets cuyos valores provienen de las instancias seleccionadas de las clases del dominio usando los *selection-Constraints* como filtros.

Estos conjuntos de datos de Awareness conforman la información requerida

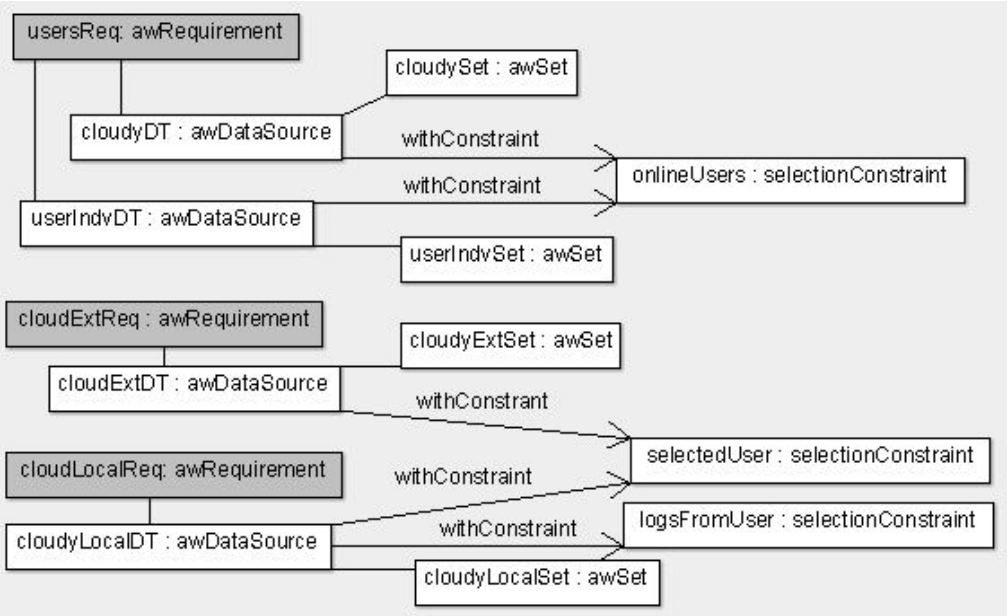


Figura 82. Awareness Requirements o más bien, awareness requeridos. Representan la información requerida por los usuarios. Estan compuestos por fuentes de datos de Awareness, los cuales a su vez están definidos estructuramente por los Awareness Sets y delimitados en acceso a datos del dominio mediante las restricciones de selección *selectionConstraints*.

por los usuarios y que está justificada en los Requerimientos de Awareness de forma informal.

La entidad *awRequirement* es un agrupador de fuentes de datos de Awareness listo para ligarse a una tarea.

En nuestro ejemplo definimos tres *awRequirement* mostrados en la Figura 82, los cuales son los siguientes:

- *usersReq*: Compuesto por las fuentes de datos *usersIndvDT* y *cloudyDT*, representa los datos de los usuarios en línea (*onlineUsers*) y la condición meteorológica de cada uno de ellos.
- *cloudyExtReq*: Compuesto por el *cloudyExtDT* representa la proyección del estado meteorológico (elemento de Awareness *cloudy weather_ zone*) de un usuario seleccionado (*selectedUser*).
- *cloudyLocalReq*: Compuesto por el *cloudyLocalDT* representa la proyección del estado meteorolgico (elemento de Awareness (*weather_ zone* y *cloudy_ local*) a partir de los datos históricos guardados en el dominio (*logsFromUser*) y relacionados con el usuario seleccionado (*selectedUser*).

Una vez definidos los Requerimientos de información se deben ligar con la tarea en donde deben ser provistos a los usuarios adecuados.

4.5. Uso de los Requerimientos de Awareness

El Awareness está ligado al proceso de toma de decisiones durante la ejecución de tareas. De forma similar, en UsiXML podemos ligar un requerimiento de Awareness a una determinada tarea junto con una condición de aplicabilidad de dicho requerimiento.

En este caso, la tarea `chat` requiere el Awareness de Usuario definido por el requerimiento *usersReq* y condicionado a que la condición *alwaysChat*, la cual

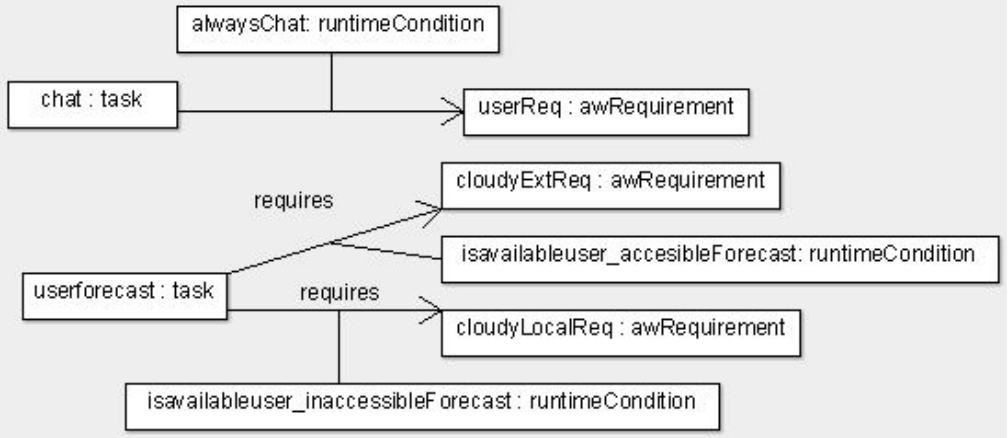


Figura 83. Definición de los requerimientos de Awareness de las tareas del sistema. Una tarea requiere ("requires") un cierto Awareness representado por un conjunto de datos dinámicos que deben ser transmitidos si la condición de aplicabilidad definida por la condición en tiempo de ejecución (*runtimeCondition*) se cumple.

representa que los datos de Awareness se van a proporcionar durante toda la tarea del `chat`.

La tarea `userforecast` tiene dos requerimientos de Awareness. El primer requerimiento (expresado mediante el enlace "requires" dice que se requiere el *cloudyExtReq* si el usuario seleccionado está disponible y si están disponibles los pronósticos meteorológicos externos. El segundo requerimiento expresa que se requiere el *cloudyLocalReq* si el usuario seleccionado está disponible para reuniones en su localidad y si es necesario utilizar los pronósticos meteorológicos locales.

Interpretación de un requerimiento hacia abajo:

La tarea `userforecast` requiere el *cloudyLocalReq* solamente si las condiciones definidas en *isavailableuser_ inaccessibleForecast* se cumplen. Esto significa que durante dicha tarea, deberán haber interfaces de usuario que observen los elementos de Awareness definidos por el Awareness Set *cloudyExtSet*. Es decir, las UIs observarán (enlace "observes") los elementos *weather_ zone* y *cloudy_ local* y proporcionarán los valores de dichos elementos en momentos futuros del tiempo como cualquier proyección. Los datos utilizados para dichos cálculos serán los

definidos por las restricciones de selección *selectedUser* y *logsFromUser*.

Las interfaces de usuario para proveer el *CloudyAw* estarán entonces enlazadas con el requerimiento de Awareness que originó dichas interfaces.

Cuando se trata de observar un elemento concreto de Awareness como el *username*, el sistema toma el valor directamente del atributo de la instancia observada. En este caso, de la instancia que representa la información de cada usuario.

Para llegar a este punto se requiere definir todas las estructuras mencionadas y disponer de los mecanismos de transformación adecuados para llevar dichos requerimientos a las interfaces de usuario finales.

Aunque el proceso de definición de los requerimientos de Awareness toma tiempo, el esfuerzo extra repercute en un ahorro de tiempo cuando se trata de agregar nuevos componentes al sistema o en el mantenimiento del mismo. El cambio es una constante en la vida de cualquier software y tener los componentes bien localizados y justificados es una ventaja que se mide en tiempo y dinero.

5. Conclusiones y trabajo futuro

A lo largo de este capítulo hemos descrito y ejemplificado la extensión al lenguaje UsiXML para incluir el soporte del Awareness a nivel de requerimientos y hemos sentado las bases para el soporte completo hasta las interfaces de usuario finales.

La extensión al metamodelo agrega el modelo del Awareness y modifica algunos modelos base para agregar algunas entidades requeridas para representar los Requerimientos de Awareness.

El modelo del Awareness permite representar cualquier tipo de Awareness que sea requerido por los usuarios. Dichos tipos de Awareness se enlazan con las entidades del dominio para darles pertenencia y significado en el propio sistema, sin

que esto limite la reutilización de estos tipos de Awareness en otros desarrollos.

El modelo de requerimiento de Awareness permite definir cada componente necesario para el soporte del Awareness a nivel de requerimiento. La definición de cada Requerimiento de Awareness comienza con la descripción general de todos los aspectos que delimitan el Awareness y su soporte para cada requerimiento, como puede ser el contexto o situación de aplicación, detalles o especificaciones sobre la interacción, etc.

El sistema de transformación de UsiXML puede ser utilizado para generar los enlaces desde los tipos de Awarenses hasta las interfaces de usuario finales. Un ejemplo de ello es el manejo de los *mappings* "observes", los cuales relacionan conceptos del dominio (un atributo, por ejemplo) con algún elemento de la interfaz de usuario abstracta o concreta.

El modelo de Requerimiento de Awareness contempla entidades que son útiles para el control de la privacidad en el acceso a datos y el control de la sobrecarga cognitiva.

Cada requerimiento expresa una necesidad de Awareness específica, así como la información que la va a satisfacer. De esta forma, los datos transmitidos a cada usuario serán especificados de acuerdo a sus necesidades de información, pero también de acuerdo a sus permisos de acceso o políticas de privacidad.

Las restricciones de acceso a los datos del sistema pueden ser dinámicas, de tal forma que puedan ser integrados mecanismos de filtrado de información, los cuales han sido utilizados para controlar la sobrecarga cognitiva derivada de las capacidades de recepción de cada usuario como individuo.

Al poder expresar requerimientos ligados a un contexto o entorno específico se pueden crear varios requerimientos de Awareness orientados a ser transmitidos a través de canales secundarios como la audición o el tacto. De esta forma se puede también reducir la sobrecarga cognitiva derivada de usar un solo canal para transmitirle los datos al usuario. Esta es una ventaja del enfoque multimodal y

multicontexto de UsiXML.

La integración de los modelos del Awareness en UsiXML extiende sus capacidades de representación sin romper su funcionamiento predefinido. En caso de no contar con sistemas de transformación que utilicen los modelos del Awareness, los modelos sirven como documentación y apoyo en el proceso de análisis y definición de requerimientos de Awareness.

Un sistema de transformación puede generar una estructura de enlaces entre los diferentes modelos de UsiXML y el modelo del Awareness, obteniendo así un código trazable en lo que respecta a mecanismos para el soporte del Awareness y los requerimientos que los especifican.

Los desarrolladores pueden tomar en cuenta el Awareness desde el principio, ya sea que usen o no los modelos. Este cambio de mentalidad ya es ganancia en lo que respecta a un desarrollo de software en el que todos los componentes deben estar representados y justificados desde las bases.

La mantenibilidad también aumenta al poder generar código a partir de especificaciones de requerimientos. En sí esta es una ventaja del desarrollo dirigido por modelos. Agregando el Awareness a este entorno deriva en más trabajo inicial de análisis y descripción, pero también deriva en mecanismos de información justificados e identificados desde las bases.

CAPITULO 6

Conclusiones generales y Trabajo Futuro

En este trabajo profundizamos en el tema de la *conciencia* (situacional o grupal) o Awareness y su relación con los sistemas colaborativos. Encontramos que el Awareness es un concepto analizado y soportado desde puntos de vista muy diversos. No obstante, se ha realizado un proceso de análisis y síntesis importante que nos ha llevado a sentar las bases para identificar sus componentes principales, su utilización y la relación que guarda con importantes procesos, como son: la interacción, la comunicación, la colaboración y la cooperación.

Al reafirmar que el Awareness es un aspecto de gran importancia para cualquier software interactivo y especialmente para el Groupware, también hemos podido constatar que el soporte del Awareness en el proceso de desarrollo del software está en sus primeras etapas y sólo disponible de forma parcial en algunas metodologías especializadas.

Después de revisar varias de las dificultades que conlleva el soporte del Awareness y su soporte actual en varias metodologías y herramientas de desarrollo, confirmamos que un aspecto imprescindible para su soporte es el modelado del propio Awareness y de sus requerimientos.

Un efecto importante de contar con modelos para caracterizar el Awareness y poder modelar los requirimientos del mismo es que pueden ser usados como base para integrar el soporte del Awareness a las metodologías de desarrollo basadas en modelos, en las cuales la trazabilidad del sistema se lleva a cabo desde las

etapas iniciales del desarrollo a las mas cercanas a los productos finales software generados, las cuales suelen estar relacionadas con el diseñ o de las interfaces de usuario.

A fin de constatar que el soporte del Awareness a nivel de requerimiento es factible en este tipo de metodologías, hemos desarrollado también una extensión a una metodología de desarrollo de sistemas interactivos basada en modelos llamada UsiXML.

Encontramos que en el diseño de un sistema Groupware hay muchas cosas que se tienen que tomar en cuenta con respecto a la colaboración, ya que un entorno virtual de trabajo impone restricciones que se encuentran en el trabajo grupal cara a cara, como son las restricciones provocadas por los medios de interacción del sistema, su diseño, sus objetivos, entre otros. El estudio del Awareness de Situación demuestra que con la experiencia y el trabajo en un entorno complejo y dinámico, los usuarios adquieren la habilidad de reconocer los elementos del entorno que realmente son necesarios y decisivos para lograr los objetivos de la tarea.

Sin embargo, durante el desarrollo de un sistema Groupware, los desarrolladores diseñan y codifican los elementos del entorno que van a estar disponibles a los usuarios y que pueden no ser suficientes para que ellos logren sus objetivos.

Esto significa que los usuarios pueden necesitar información extra que nunca les ha sido proporcionada, pero no lo saben. Soólo saben que lograr sus objetivos es muy difícil o imposible, pero no saben porqué. Si el entorno te da 20 elementos de Awareness pero el sistema soólo te transmite 5 y en los 15 que faltan estaban los más importantes, va a ser muy difícil que ellos se den cuenta de esta carencia ya que tal vez nunca han tenido experiencia previa en el uso de los mecanismos proporcionados por los entornos colaborativos cara a cara.

Por consiguiente, es necesario estudiar más profundamente los entornos colaborativos que queremos soportar ya que en un ambiente sin intermediarios de

software la información fluye por muchos canales y en grandes cantidades de forma paralela. Mientras que en los sistemas Groupware habitualmente no es así.

Mejorar la colaboración en el Groupware depende en gran medida de que se aplique el conocimiento y la experiencia que hay sobre el trabajo colaborativo sin intermediarios y que se abran más canales de comunicación usuario-computadora que permitan reducir las barreras de interacción existentes ahora mismo y que incrementan las dificultades inherentes a la generación y uso del Awareness.

Casi todos los software soportan algo de Awareness, pero no de forma organizada, justificada y fácilmente mantenible. De hecho, el soporte del Awareness se reduce al soporte de la *retroalimentación* o *feedback*[1] como base para la interacción, sin tener en cuenta que el Awareness, como estado mental, es la verdadera herramienta para tomar decisiones y poder interactuar con eficacia. Pensamos que en este camino de evolución en el desarrollo de software, el Awareness es un aspecto que todavía necesita integrarse mejor a los sistemas informáticos y a su proceso de desarrollo, ya que la tendencia es soportar cada vez más procesos de trabajo usando sistemas informáticos.

Esperamos que este trabajo sea de utilidad para científicos y desarrolladores interados en mejorar los procesos interactivos y el trabajo en grupo de sus sistemas. Como dice una de las máximas de la Usabilidad según [Lund, 1997]: "*La información para la decisión debe estar ahí cuando se requiere tomar la decisión*". A lo largo de este trabajo podemos afirmar que la información a la que se refiere Lund es el Awareness.

1. Aportaciones

El fruto de esta tesis doctoral se resume en las siguientes aportaciones:

- Una revisión sobre varias metodologías de desarrollo específicas para sis-

[1] `http://en.wikipedia.org/wiki/Feedback`

temas Groupware resaltando los aspectos relacionados con el soporte de la conciencia de grupo o Awareness y el trabajo grupal, encontrando puntos de mejora en la representación y seguimiento de los requerimientos de información de Awareness.

- Se han analizado las extensas aportaciones existentes sobre la aplicación, importancia y soporte del Awarenes y se ha realizado un proceso de síntesis para facilitar la caracterización general de los procesos de Awareness desde las fases de elicitación de requerimientos hasta su soporte final en las interfaces de usuario. Podemos resaltar los siguientes puntos:

  - Una amplia revisión de distintas definiciones del término Awareness desde distintas perspectivas y entornos de uso.
  - El estudio de la relación entre el Awareness y los procesos de interacción y colaboración soportados por ordenador.
  - Un análisis recopilatorio de varias técnicas para el soporte del la conciencia grupal y el Awareness en distintas fases del desarrollo de un sistema.
  - Una revisón de las dificultades derivadas del soporte del Awareness y de varias técnicas para reducir sus efectos.
  - Se incluye una extensa recopilación y descripción de varios tipos de Awareness encontrados en la literatura científica, especialmente en la referente a los sistemas Groupware.
  - Un estudio sobre las formas de transmisión de la información de Awareness a través de las interfaces de usuario. Se analizan los mecanismos de transmisión de información, los canales de recepción de información disponibles en el ser humano y se recopilan diferentes interfaces de usuario especializadas en proporcionar Awareness.

- Un modelo del Awareness para representar sus elementos y tipos a nivel conceptual y orientado al desarrollo de software. De esta manera será posi-

ble representar cualquier tipo de Awareness y expresar de forma concreta y enlazable la información que servirá para generar el estado de conocimiento conocido como *Conciencia* o *Awareness*.

- La capacidad de representar cualquier tipo de Awareness, lo cual permite cubrir todo el espectro de los posibles requerimientos de Awareness que pueda tener un sistema Groupware, sin limitarse a algún tipo de Awareness específico como por ejemplo el *Workspace Awareness*, el *Activity Awareness*, entre otros.

- Un modelo para la representación de los requerimientos de Awareness, el cual permite representar las condiciones y restricciones bajo las cuales se debe proporcionar información de Awareness a un determinado grupo de usuarios durante una situación y/o contexto específico.

- Un marco de referencia para describir el soporte del Awareness dentro de una metodología de desarrollo. Este marco de referencia puede utilizarce, tanto para guiar la integración del Awareness en una metodología de desarrollo, como para analizar el nivel de soporte del Awareness proporcionado por una metodología existente.

- Una extensión al lenguaje UsiXML para el soporte del Awareness a nivel de requisitos, proporcionando las bases para el soporte completo hasta las interfaces de usuario finales. Dicha extensión incrementa la capacidad representativa de UsiXML y le permite tomar en cuenta el Awareness desde las primeras fases de desarrollo, a fin de aprovechar dicha información una vez que se tengan procesos de transformación y modelos a nivel de diseño más potentes que tomen en cuenta el Awareness para generar las interfaces de usuario.

- Una nueva propuesta sobre cómo abordar el Awareness requerido en un sistema y soportado por una metodología de desarrollo. Esta nueva propuesta se centra en capturar las necesidades de información de los usuarios, inde-

pendientemente del tipo de sistema o forma de trabajo. El Awareness puede requerirse tanto en tareas colaborativas y cooperativas como en tareas individuales, por lo que enfocarse en soportar un sólo tipo de Awareness solo limita al propio sistema y puede dejar a los usuarios con necesidades sin cubrir.

- Una investigación científica orientada a la mejora de los sistemas Groupware en la cual se concentran, analizan y sintetizan diversos trabajos para apoyar el soporte completo del Awareness durante el desarrollo y mantenimiento del sistema siguiendo un enfoque dirigido por modelos. Los trabajos estudiados y los aportes realizados sirven como guía para mejorar el soporte del trabajo grupal y los procesos altamente interactivos que pueden presentarse en los sistemas de software actuales.

La producción científica relacionada con el presente documento es la siguiente:

- Figueroa Martínez José, Gutierrez Vela F. L., Collazos C. A.: "**Awareness Models for the Development of Ubiquitous Systems**", Juan Carlos Augusto, Juan M. Corchado, Paulo Novais, and Cesar Analide (Eds.), International Symposium on Ambient Intelligence (ISAmI 2010), Volumen 72, pag. 237-245, Springer, 2010

  En este trabajo se presentan los primeros modelos conceptuales del Awareness así como un estudio del Awareness y su importancia en los sistemas ubicuos. Publicado en un volumen especial de la serie Advanced in Intelligent and Soft Computing (Springer Verlag), el cual está indexado por ISI Conference Proceedings Citation Index - Science (CPCI-S), incluido en ISI Web of Science, SCOPUS , DBLP, Ulrich's, EI-Compendex, Zentralblatt Math, MetaPress y Springerlink entre otros.

- Figueroa Martínez José, Gutierrez Vela F.L.: "**Modelado del Awareness para el Desarrollo de Sistemas Colaborativos.**", Congreso Interacio-

nal de Interacción Persona-Ordenador INTERACCION2010, Valencia, España, 2010

En este trabajo se presentan los primeros modelos conceptuales del Awareness orientados a describir la información de Awareness necesaria para soportar el trabajo grupal.

- Figueroa-Martínez José , Gutierrez Vela F.L., López-Jaquero Víctor, González Pascual: "**Extensión a UsiXML para el Soporte del Awareness**", Congreso Interacional de Interacción Persona-Ordenador INTERACCION2011, pag. 1-9, Lisboa, Portugal, 2011

  En este trabajo se presenta una mejora de los modelos del Awareness integrados al lenguaje UsiXML a nivel de requisitos.

- Figueroa-Martínez José , Gutierrez Vela F.L., López-Jaquero Víctor , González Pascual: "**UsiXML extension for Awareness Support**", 13th IFIP (CORE A) TC13 Conference on Human-Computer Interaction INTERACT2011, volumen 6949, pag. 665-668, Lisboa, Portugal, 2011

  En este trabajo se presentan la extensión al lenguaje UsiXML para el soporte del Awareness desde el punto de vista del apoyo al desarrollo de sistemas interactivos, multimodales y multicontexto, centrándose en el soporte del Awareness como aspecto esencial para la colaboración y la interacción. Las actas del congreso están indexadas en *CORE tipo A* y publicadas en un volumen especial de la serie Lecture Notes (Springer Verlag). Indexado por ISI Conference Proceedings Citation Index - Science (CPCI-S), incluido en ISI Web of Science, SCOPUS, EI Engineering Index (Compendex and Inspec databases), ACM Digital Library, DBLP, Springerlink, entre otros.

- Figueroa-Martínez J., López Jaquero V., Gutierrez Vela F.L., González P.: "**Enriching UsiXML language to support Awareness requirements**", Journal of Science of Computer Programming *en revisión*, 2012

  En este trabajo se presenta una versión extendida y fundamentada concep-

tualmente de la extensión a UsiXML para el soporte del Awareness a nivel de requisitos. Se presentan los modelos del Awareness y de requerimientos de Awareness integrados a la metodología, así como el marco de referencia utilizado para su desarrollo y un proceso de validación de los modelos.

La revista es editada por Elsevier e incluida en JCR ISI en la categoría *Computer Science* con un índice de impacto de **1.282**. El trabajo fue seleccionado del congreso INTERACCION'11, extendido y en proceso de evaluación por el comité editorial de la revista. Actualmente se encuentra en la 3 ronda de revisión.

Producción científica complementaria durante el periodo doctoral:

- Figueroa-Martínez José, Morales Lluvia, Castillo Luis: "**Modificación del Módulo SCORM para el Soporte Básico del IMS-LD**", Moodle Moot 2008 (Barcelona, España), 2008
- Castillo Luis, Fernandez-Olivares Juan, González Antonio, Milla Gonzalo, Prior David, Morales Lluvia, Figueroa José, Pérez-Villar Victor: "**A Knowledge Engineering Methodology for Rapid Prototyping of Planning Applications**", 23rd International FLAIRS Conference, Daytona Beach, Florida, USA, 2010

2. Trabajo futuro

Este trabajo ha abierto varias posibilidades de investigación en lo que respecta al papel de la integración del Awareness en el desarrollo de software.

En primer lugar, la extensión al lenguaje UsiXML puede completarse y llevarse a niveles más bajos de abstracción hasta las interfaces de usuario finales. Este proceso requiere definir reglas de transformación entre los modelos de Awareness propuestos, sus requerimientos y los mecanismos usados a nivel de IUs, para así lograr el soporte completo del Awareness en UsiXML.

Aunque la extensión descrita en este capítulo es para la versión 1.8 de UsiXML, estamos colaborando con miembros del comité de estandarización de UsiXML para incluir los modelos del Awareness adaptados al nuevo metamodelo de UsiXML.

Varias metodologías de desarrollo pueden beneficiarse de las herramientas conceptuales propuestas en este trabajo, por lo cual se pretende trabajar estrechamente con los diseñadores de algunas metodologías orientadas al Groupware para integrar los modelos conceptuales del Awareness al proceso de desarrollo.

Partiendo de los trabajos de catalogación de mecanismos de colaboración a nivel conceptual propuestos por José Luis Isla [Isla Montes, 2007] y utilizando el "*profile*" [Isla et al., 2005] para la definición de patrones conceptuales, podríamos definir patrones que modelen los tipos de Awareness existentes, por medio de un catálogo de Tipos de Awareness organizados por uso y contexto puede ser muy beneficioso para los diseñadores de sistemas interactivos, de tal forma que pueda reutilizarse y centralizarse la experiencia ganada durante la identificación y definición de Tipos de Awareness.

Las interfaces de usuario para el Awareness son también un aspecto muy importante del soporte del Awareness, ya que, con ayuda de los modelos, pueden estár directamente ligadas a las fuentes de datos definidas por los Tipos de Awareness, además de estár justificadas por los requerimientos de Awareness, lo cual permite validar el uso de dichas interfaces.

A nivel general, se puede elaborar también un catálogo de interfaces de usuario para Awareness así como un conjunto de guías específicas para diseñar y combinar interfaces de Awareness que permitan la interacción en determinados entornos.

A lo largo de la tesis nos hemos planteado como objetivo definir una forma de representar los requisitos de Awareness y darle soporte al proceso de desarrolo, pero consideramos que en esta área el enfoque propuesto sobre la importancia

del Awareness puede abrir nuevos caminos en lo que respecta a la obtención y manejo de los requerimientos de Awareness.

Como puede verse, se han abierto varios caminos en los que el Awareness es un participante activo, con lo cual es cuestión de elegir y seguir trabajando en algunos de ellos.

Bibliografía

[Ahmed and Brahim, 2008] Ahmed, K. and Brahim, B. (2008). Towards a web based simulation groupware: Experiment with bscw. *WSEAS Transactions on Business Economics*, 5(1).

[Baldauf et al., 2007] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing IJAHUC*, 2(4).

[Bales et al., 2011] Bales, E., Li, K. A., and Griwsold, W. (2011). Couplevibe: Mobile implicit communication to improve awareness for (long-distance) couples. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, CSCW '11, pages 65–74, New York, NY, USA. ACM.

[Balsters, 2003] Balsters, H. (2003). Modelling database views with derived classes in the uml/ocl-framework. In Stevens, P., Whittle, J., and Booch, G., editors, *UML 2003 - The Unified Modeling Language. Modeling Languages and Applications*, volume 2863 of *LNCS*, pages 295–309. Springer.

[Baumgartner et al., 2010] Baumgartner, N., Gottesheim, W., Mitsch, S., Retschitzegger, W., and Schwinger, W. (2010). Beaware!-situation awareness, the ontology-driven way. *Data & Knowledge Engineering*, 69(11):1181–1193.

[Begole and Tang, 2007] Begole, J. B. and Tang, J. C. (2007). Incorporating human and machine interpretation of unavailability and rhythm awareness into the design of collaborative applications. *Human-Computer Interaction*, 22(1):7–45.

[Benyon and Murray, 1993] Benyon, D. and Murray, D. (1993). Developing adaptive systems to fit individual aptitudes. In *Proceedings of the 1st international conference on Intelligent user interfaces*, IUI '93, pages 115–121, New York, NY, USA. ACM.

[Braune and Trollip, 1982] Braune, R. J. and Trollip, S. R. (1982). Towards and interal model in pilot training. *Avation, Space and Environmental Medicine*, 53:996–999.

[Cadiz et al., 2001] Cadiz, J., Venolia, G. D., Jancke, G., and Gupta, A. (2001). Sideshow: Providing peripheral awareness of important information. Technical report, Microsoft Research, Collaboration, and Multimedia Group.

[Calvary et al., 2003] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289 – 308. Computer-Aided Design of User Interface.

[Carroll et al., 2003] Carroll, J. M., Neale, D. C., Isenhour, P. L., Rosson, M. B., and McCrickard, D. (2003). Notification and awareness: synchronizing task-oriented collaborative activity. *International Journal of Human-Computer Studies*, 58(5):605 – 632.

[Carroll et al., 2006] Carroll, J. M., Rosson, M. B., Convertino, G., and Ganoe, C. H. (2006). Awareness and teamwork in computer-supported collaborations. *Interacting with Computers*, 18(1):21–46.

[Carter and Dewan, 2010] Carter, J. and Dewan, P. (2010). Are you having difficulty? In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW 2010, pages 211–214, New York, NY, USA. ACM.

[Castillo et al., 2010] Castillo, L., Fernández-Olivares, J., González, A., Milla, G., Prior, D., Morales, L., Figueroa, J., and Pérez-Villar, V. (2010). A knowledge engineering methodology for rapid prototyping of planning applications. In *FLAIRS Conference*.

[Castro et al., 2001] Castro, J., Kolp, M., and Mylopoulos, J. (2001). A requirements-driven development methodology. In Dittrich, K., Geppert, A., and Norrie, M., editors, *Advanced Information Systems Engineering*, volume 2068 of *Lecture Notes in Computer Science*, pages 108–123. Springer Berlin / Heidelberg.

[Collazos et al., 2003] Collazos, C., Guerrero, L., and Pino, J. (2003). Knowledge construction awareness. *Journal of Student Centered Learning*, 1(2):76–86.

[Coninx et al., 2003] Coninx, K., Luyten, K., Vandervelpen, C., Van den Bergh, J., and Creemers, B. (2003). Dygimes: Dynamically generating interfaces for mobile computing devices and embedded systems. In Chittaro, L., editor, *Human-Computer Interaction with Mobile Devices and Services*, volume 2795 of *Lecture Notes in Computer Science*, pages 256–270. Springer Berlin / Heidelberg. 10.1007/978-3-540-45233-1_19.

[Cook et al., 2009] Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277 – 298.

[David and Borges, 2001] David, J. and Borges, M. (2001). Selectivity of awareness components in asynchronous cscw environments. In *Groupware, 2001. Proceedings. Seventh International Workshop on*, pages 115 –124.

[Dillenbourg et al., 1996] Dillenbourg, P., Baker, M., Blaye, A., and O' Malley, C. (1996). The evolution of research on collaborative learning. In Spada, E. and Reiman, P., editors, *Learning in Humans and Machine: Towards an interdisciplinary learning science.*, pages 189–211. Oxford: Elsevier.

[Dourish and Bellotti, 1992] Dourish, P. and Bellotti, V. (1992). Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, CSCW '92, pages 107–114, New York, NY, USA. ACM.

[Dourish and Bly, 1992] Dourish, P. and Bly, S. (1992). Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 541–547, New York, NY, USA. ACM.

[Drury and Williams, 2002] Drury, J. and Williams, M. G. (2002). A framework for role-based specification and evaluation of awareness support in synchronous collaborative applications. In *Proceedings of the 11th IEEE International Workshops on Enabling Technologies: nfrastructure for Collaborative Enterprises*, pages 12–17, Washington, DC, USA. IEEE Computer Society.

[Drury et al., 2006] Drury, J. L., Riek, L., and Rackliffe, N. (2006). A decomposition of uav-related situation awareness. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, HRI '06, pages 88–94, New York, NY, USA. ACM.

[Endsley, 1995] Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37:32–64(33).

[Endsley, 2001] Endsley, M. R. (2001). Designing for situation awareness in complex systems. In *Proceedings of the Second intenational workshop on symbiosis of humans, artifacts and environment*, Kyoto, Japan.

[Endsley et al., 2003] Endsley, M. R., Bolstad, C. A., Jones, D. G., and Riley, J. M. (2003). Situation awareness oriented design: From user's cognitive requirements to creating effective supporting technologies. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 47, pages 268–272.

[Farooq et al., 2007] Farooq, U., Carroll, J. M., and Ganoe, C. H. (2007). Supporting creativity with awareness in distributed collaboration. In *Proceedings of the 2007 international ACM conference on Supporting group work*, GROUP '07, pages 31–40, New York, NY, USA. ACM.

[Feng et al., 2009] Feng, Y., Teng, T., and Tan, A. (2009). Modelling situation awareness for context-aware decision support. *Expert Systems with Applications*, 36(1):455–463.

[Fracker, 1989] Fracker, M. L. (1989). Attention gradients in situation awareness. In *Situational awareness in aerospace operations (AGARD-CP-478)*, pages 6/1–6/10, Neuilly-Sur-Seine, France. NATO-Advisory Group for Aerospace Research and Development.

[Gallego et al., 2011] Gallego, F., Molina, A. I., Gallardo, J., and Bravo, C. (2011). A conceptual framework for modeling awareness mechanisms in collaborative systems. In Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., and Winckler, M., editors, *Human-Computer Interaction INTERACT 2011*, volume 6949 of *Lecture Notes in Computer Science*, pages 454–457. Springer Berlin / Heidelberg. 10.1007/978-3-642-23768-3_108.

[Garrido et al., 2002] Garrido, J. L., Gea, M., Padilla, N., Cañas, J. J., and Waern, Y. (2002). Amenities: Modelado de entornos cooperativos. In *Actas del III Congreso Internacional de Interacción Persona-Ordenador 2002*, pages 97–104, Madrid, España, Mayo 2002.

[Garrido Bullejos, 2003] Garrido Bullejos, J. L. (2003). *AMENITIES: Una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tareas.* PhD thesis, Universidad de Granada, Granada, Spain.

[Goodrich et al., 2006] Goodrich, K., Schutte, P., Flemisch, F., and Williams, R. (2006). Application of the h-mode, a design and interaction concept for highly automated vehicles, to aircraft. In *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, pages 1 –13.

[Group, 2001] Group, O. M. (2001). Unified modelling language specification.

[Guerrero García et al., 2009] Guerrero García, J., González-Calleros, J. M., Vanderdonckt, J., and Muñoz Arteaga, J. (2009). heoretical survey of user

interface description languages: Preliminary results. In *Proceedings of the Web Congress, 2009. LA-WEB '09. Latin American*, pages 36–43.

[Gutiérrez et al., 2006] Gutiérrez, F. L., Penichet, V. M. R., Isla, J., Montero, F., Lozano, M. D., Gallud, J. A., and Rodríguez, L. (2006). Un marco conceptual para el modelado de sistemas colaborativos empresariales. In *Proceedings del VII Congreso Internacional de Interacción Persona-Ordenador - Interacción2006*, Universidad de Castilla-La Mancha, Puertollano, Ciudad Real, Spain.

[Gutwin and Greenberg, 2002] Gutwin, C. and Greenberg, S. (2002). A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3):411–446. 10.1023/A:1021271517844.

[Gutwin and Greenberg, 2004] Gutwin, C. and Greenberg, S. (2004). The importance of Awareness for team cognition in distributed Collaboration. *Team cognition Understanding the factors that drive process and performance*, 201(2001-696-19):1–33.

[Gutwin et al., 2008] Gutwin, C., Greenberg, S., Blum, R., Dyck, J., Tee, K., and Mcewan, G. (2008). Supporting informal collaboration in shared-workspace groupware. *Journal of Universal Computer Science*, 14(9):1411–1434.

[Gutwin et al., 1996a] Gutwin, C., Greenberg, S., and Roseman, M. (1996a). Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. In *Proceedings of HCI on People and Computers XI*, pages 281–298, London, UK. Springer-Verlag.

[Gutwin et al., 1996b] Gutwin, C., Greenberg, S., and Roseman, M. (1996b). Workspace awareness support with radar views. In *Conference companion on Human factors in computing systems: common ground*, CHI 1996, pages 210–211, New York, NY, USA. ACM.

[Gutwin et al., 1995] Gutwin, C., Stark, G., and Greenberg, S. (1995). Support for workspace awareness in educational groupware. In *The first international conference on Computer support for collaborative learning*, CSCL 1995, pages 147–156, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

[Hailpern and Tarr, 2006] Hailpern, B. and Tarr, P. (2006). Model-driven development: The good, the bad, and the ugly. *IBM Systems Journal*, 45(3):451–461.

[Hill and Gutwin, 2004] Hill, J. and Gutwin, C. (2004). The maui toolkit: Groupware widgets for group awareness. *Computer Supported Cooperative Work (CSCW)*, 13:539–571. 10.1007/s10606-004-5063-7.

[Hinze-Hoare, 2006] Hinze-Hoare, V. (2006). Cscr:computer supported collaborative research. *CoRR*, abs/cs/0611042.

[Hockey, 1970] Hockey, G. R. J. (1970). Effect of loud noise on attentional selectivity. *Quarterly Journal of Experimental Psychology*, 22:28–36.

[Hockey, 1986] Hockey, G. R. J. (1986). Changes in operator efficiency as a function of environmental stress, fatigue and circadian rhythms. In Boff, K., Kaufman, L., and Thomas, J., editors, *Handbook of perception and human performance*, pages 44/1–44/49, New York. Wiley.

[Holmquist et al., 1999] Holmquist, L., Falk, J., and Wigström, J. (1999). Supporting group collaboration with interpersonal awareness devices. *Personal and Ubiquitous Computing*, 3(1):13–21. 10.1007/BF01305316.

[Humphreys, 1981] Humphreys, G. W. (1981). Flexibility of attention between stimulus dimensions. *Perception and Psychophysics*, 30:291–302.

[Ishii, 2008] Ishii, H. (2008). The tangible user interface and its evolution. *Commun. ACM*, 51(6):32–36.

[Isla et al., 2005] Isla, J. L., Gutierrez, F., and Paderewski, P. (2005). Un profile

para el modelado de patrones de software. jornadas de ingenieria del software y bases de datos. In *Procs. Of the JISBD.05-CEDI.05*, pages 265–270.

[Isla Montes, 2007] Isla Montes, J. L. (2007). *Modelado Conceptual de Sistemas Cooperativos en base a Patrones en Amenities.* PhD thesis, Universidad de Granada.

[Jensen, 1996] Jensen, K. (1996). *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use.* Springer, 2nd edition.

[Johnson, 1989] Johnson, P. (1989). *Supporting system design by analyzing current task knowledge*, pages 160–185. Wiley.

[Kim and Kim, 2007] Kim, M.-K. and Kim, H.-C. (2007). How to build awareness-supported systems without sacrificing privacy. In Shen, W., Luo, J., Lin, Z., Barthes, J.-P., and Hao, Q., editors, *Computer Supported Cooperative Work in Design III*, volume 4402 of *Lecture Notes in Computer Science*, pages 609–618. Springer Berlin / Heidelberg.

[Kimball and Ross, 2002] Kimball, R. and Ross, M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* Wiley, 2ed edition.

[Kirsch-Pinheiro et al., 2005] Kirsch-Pinheiro, M., Villanova-Oliver, M., Gensel, J., and Martin, H. (2005). Bw-m: a framework for awareness support in web-based groupware systems. In *Computer Supported Cooperative Work in Design, 2005. Proceedings of the Ninth International Conference on*, volume 1, pages 240 – 246.

[Krause et al., 2006] Krause, A., Smailagic, A., and Siewiorek, D. (2006). Context-aware mobile computing: learning context- dependent personal preferences from a wearable sensor array. *Mobile Computing, IEEE Transactions on*, 5(2):113 – 127.

[Kwon et al., 2011] Kwon, G. H., Lee, Y. S., and Kumar, M. (2011). The tree of knowledge: a localized collective intelligence tool. In *Proceedings of the ACM*

2011 conference on Computer supported cooperative work, CSCW '11, pages 665–668, New York, NY, USA. ACM.

[Liechti, 2000] Liechti, O. (2000). Awareness and the www: an overview. *SIGGROUP Bull.*, 21(3):3–12.

[Limbourg et al., 2005] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lóez-Jaquero, V. (2005). Usixml: A language supporting multi-path development of user interfaces. In Bastide, R., Palanque, P., and Roth, J., editors, *Engineering Human Computer Interaction and Interactive Systems*, volume 3425 of *Lecture Notes in Computer Science*, pages 134–135. Springer. 10.1007/11431879_12.

[Lund, 1997] Lund, A. M. (1997). Expert ratings of usability maxims. *Ergonomics in Design: The Quarterly of Human Factors Applications*, 5(3):15–20.

[Luqman and Griss, 2010] Luqman, F. and Griss, M. (2010). Overseer: A mobile context-aware collaboration and task management system for disaster response. In *Procs. of the 2010 Eighth International Conference on Creating, Connecting and Collaborating through Computing*, pages 76–82. IEEE Computer Society.

[Matheus et al., 2003] Matheus, C., Kokar, M., and Baclawski, K. (2003). A core ontology for situation awareness. In *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, volume 1, pages 545 – 552.

[Matthews et al., 2007] Matthews, T., Rattenbury, T., and Carter, S. (2007). Defining, designing, and evaluating peripheral displays: an analysis using activity theory. *Human-Computer Interaction*, 22(1):221–261.

[Mayer, 1983] Mayer, R. E. (1983). *Thinking, problem solving, cognition.* Freeman, New York.

[Mellor et al., 2003] Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Guest editorsíntroduction: Model-driven development. *IEEE Software*, 20:14–18.

[Molina et al., 2006] Molina, A., Redondo, M., and Ortega, M. (2006). A conceptual and methodological framework for modeling interactive groupware applications. In Dimitriadis, Y., Zigurs, I., and Gómez-Sánchez, E., editors, *Groupware: Design, Implementation, and Use*, volume 4154 of *Lecture Notes in Computer Science*, pages 413–420. Springer.

[Molina et al., 2009] Molina, A. I., Redondo, M. A., and Ortega, M. (2009). A methodological approach for user interface development of collaborative applications: A case study. *Science of Computer Programming*, 74(9):754 – 776.

[Molina et al., 2008] Molina, A. I., Redondo, M. A., Ortega, M., and Hoppe, U. (2008). Ciam: A methodology for the development of groupware user interfaces. *Journal of Universal Computer Science*, 14(9):1435–1446.

[Montero et al., 2006] Montero, F., López-Jaquero, V., Vanderdonckt, J., González, P., Lozano, M., and Limbourg, Q. (2006). Solving the mapping problem in user interface design by seamless integration in idealxml. In Gilroy, S. and Harrison, M., editors, *Interactive Systems. Design, Specification, and Verification*, volume 3941 of *Lecture Notes in Computer Science*, pages 161–172. Springer Berlin / Heidelberg. 10.1007/11752707_14.

[Morán et al., 2001] Morán, A. L., Favela, J., Enríquez, A. M. M., and Decouchant, D. (2001). Document presence notification services for collaborative writing. In *Groupware, 2001. Proceedings. Seventh International Workshop on Groupware*, pages 125 –133, Washington, DC, USA. IEEE Computer Society.

[Mori et al., 2004] Mori, G., Paterno, F., and Santoro, C. (2004). Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, 30(8):507–520.

[Neyem et al., 2007] Neyem, A., Aracena, C., Collazos, C. A., and Alarcón, R. (2007). Designing emotional awareness devices: what one sees is what one feels. *Revista Chilena de Ingeniería*, 15(3):227–235.

[Ogata et al., 1996] Ogata, H., Matsuura, K., and Yano, Y. (1996). Knowledge awareness: Bridging between shared knowledge and collaboration in sharlok. In *Proc. Of Ed-Media 1996*.

[Ogata and Yano, 2000] Ogata, H. and Yano, Y. (2000). Combining knowledge awareness and information filtering in an open-ended collaborative learning environment. *International Journal of Artificial Intelligence in Education*, 11:33–46.

[Paternò, 1999] Paternò, F. (1999). *ConcurTaskTrees : An Engineered Approach to Model-based Design of Interactive Systems*, volume 25, pages 1–18. Lawrence Erlbaum Associates.

[Paterno, 2000] Paterno, F. (2000). Model-based design of interactive applications. *Intelligence*, 11(4):26–38.

[Paternò, 2004] Paternò, F. (2004). *ConcurTaskTrees: An Engineered Notation for Task Models*, pages 483–503. Mahwah: Lawrence Erlbaum.

[Paternò et al., 2001] Paternò, F., Mori, G., and Galiberti, R. (2001). Ctte: an environment for analysis and development of task models of cooperative applications. In *CHI '01 extended abstracts on Human factors in computing systems*, CHI EA '01, pages 21–22, New York, NY, USA. ACM.

[Pedersen and Sokoler, 1997] Pedersen, E. R. and Sokoler, T. (1997). Aroma: abstract representation of presence supporting mutual awareness. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, pages 51–58, New York, NY, USA. ACM.

[Penichet et al., 2009a] Penichet, V. M., Lozano, M. D., Gallud, J. A., and Tesoriero, R. (2009a). Requirement gathering templates for groupware applications. In Macías, J. A., Granollers Saltiveri, A., and Latorre, P. M., editors, *New Trends on Human-Computer Interaction*, pages 1–10. Springer London.

[Penichet et al., 2009b] Penichet, V. M., Lozano, M. D., Gallud, J. A., and Tesoriero, R. (2009b). User interface analysis for groupware applications in the touche process model. *Advances in Engineering Software*, 40(12):1212 – 1222. Designing, modelling and implementing interactive systems.

[Penichet et al., 2010] Penichet, V. M., Lozano, M. D., Gallud, J. A., and Tesoriero, R. (2010). Requirement-based approach for groupware environments design. *Journal of Systems and Software*, 83(8):1478 – 1488.

[Power and Sharda, 2009] Power, D. J. and Sharda, R. (2009). Decision support systems. In Nof, S. Y., editor, *Springer Handbook of Automation*, pages 1539–1548. Springer Berlin Heidelberg.

[Prinz, 2002] Prinz, W. (2002). Nessie: An awareness environment for cooperative settings. In Bodker, S., Kyng, M., and Schmidt, K., editors, *ECSCW .99*, pages 391–410. Springer Netherlands.

[Prinz et al., 2010] Prinz, W., Hinrichs, E., and Kireyev, I. (2010). Anticipative awareness in a groupware system. In Randall, D. and Salembier, P., editors, *From CSCW to Web 2.0: European Developments in Collaborative Design*, Computer Supported Cooperative Work, pages 3–20. Springer London.

[Richman and Slovak, 1987] Richman, L. S. and Slovak, J. (1987). Software catches the team spirit new computer programs may soon change the way groups of people work together – and start delivering the long-awaited payoff from office automation. *FORTUNE Magazine*.

[Riener and Ferscha, 2008] Riener, A. and Ferscha, A. (2008). Raising awareness about space via vibro-tactile notifications. In Roggen, D., Lombriser, C., Tröster, G., Kortuem, G., and Havinga, P., editors, *Smart Sensing and Context*, volume 5279 of *Lecture Notes in Computer Science*, pages 235–245. Springer.

[Ritchie and Thompson, 1974] Ritchie, D. M. and Thompson, K. (1974). The unix time-sharing system. *Commun. ACM*, 17:365–375.

[Roseman and Greenberg, 1996] Roseman, M. and Greenberg, S. (1996). Building real-time groupware with groupkit, a groupware toolkit. *ACM Trans. Comput.-Hum. Interact.*, 3(1):66–106.

[Rosenberg, 2001] Rosenberg, M. J. (2001). *E-learning: Estrategias para Transmitir Conocimiento en la Era Digital*. Mc Graw Hill.

[Rouse and Morris, 1985] Rouse, W. B. and Morris, N. M. (1985). On looking into the black box: Prospects and limits in the search for mental models. Technical Report DTIC AD-A159080, Georgia Institute of Technology, Center for Man-Machine Systems Research, Atlanta, Georgia.

[Ruiz Penichet, 2007] Ruiz Penichet, V. M. (2007). *Modelo de Proceso para el Desarrollo de Interfaces en Entornos CSCW Centrado en los Usuarios y Dirigido por Tareas*. PhD thesis, Universidad de Castilla-La Mancha, Departamento de Sistemas Informáticos.

[Rumbaugh et al., 1999] Rumbaugh, J., Jacobson, I., and Booch, G. (1999). *The Unified Modeling Language - Reference Manual*. Addison-Wesley.

[Sahami et al., 2008] Sahami, A., Holleis, P., Schmidt, A., and Häkkilä, J. (2008). Rich tactile output on mobile devices. In Aarts, E., Crowley, J., de Ruyter, B., Gerhäuser, H., Pflaum, A., Schmidt, J., and Wichert, R., editors, *Ambient Intelligence*, volume 5355 of *Lecture Notes in Computer Science*, pages 210–221. Springer.

[Schmidt, 2002] Schmidt, K. (2002). The problem with áwareness': Introductory remarks on áwareness in cscw'. *Comput. Supported Coop. Work*, 11(3):285–298.

[Sharit and Salvendy, 1982] Sharit, J. and Salvendy, G. (1982). Occupational stress: Review and reappraisal. *Human Factors*, 24:129–162.

[Sohlenkamp, 1999] Sohlenkamp, M. (1999). Supporting group awareness in

multi-user environments through perceptualization. Technical Report 6, European Research Consortium for Informatics and Mathematics.

[Strang and Popien, 2004] Strang, T. and Popien, C. L. (2004). A context modeling survey. In *Procs. of The Sixth International Conference on Ubiquitous Computing, Workshop on Advanced Context Modelling, Reasoning and Management.*

[Tam and Greenberg, 2006] Tam, J. and Greenberg, S. (2006). A framework for asynchronous change awareness in collaborative documents and workspaces. *International Journal of Human-Computer Studies*, 64(7):583–598.

[Tecuci et al., 2007] Tecuci, G., Boicu, M., and Cox, M. T. (2007). Seven aspects of mixed-initiative reasoning: An introduction to this special issue on mixed-initiative assistants. *AI Magazine*, 28(2).

[Teruel et al., 2011] Teruel, M., Navarro, E., López-Jaquero, V., Montero, F., and González, P. (2011). Csrml: A goal-oriented approach to model requirements for collaborative systems. In Jeusfeld, M., Delcambre, L., and Ling, T.-W., editors, *Conceptual Modeling - ER 2011*, volume 6998 of *LNCS*, pages 33–46. Springer Berlin / Heidelberg. 10.1007/978-3-642-24606-7_4.

[Treisman and Paterson, 1984] Treisman, A. and Paterson, R. (1984). Emergent features, attention and object perception. *Journal of Experimenta Psychology: Human Perception and Performance*, 10:12–31.

[Tsai et al., 2010] Tsai, F. S., Etoh, M., Xie, X., Lee, W.-C., and Yang, Q. (2010). Introduction to mobile information retrieval. *IEEE Intelligent Systems*, 25(1):11–15.

[Tzovaras, 2008] Tzovaras, D., editor (2008). *Multimodal User Interfaces*. Signals and Communication Technology. Springer.

[Vertegaal et al., 1997] Vertegaal, R., Velichkovsky, B., and van der Veer, G.

(1997). Catching the eye: management of joint attention in cooperative work. *SIGCHI Bull.*, 29:87–92.

[Wickens, 1984] Wickens, C. D. (1984). *Engineering psychology and human performance*. Merrill, Columbus, OH, 1st edition.

[Wickens, 1992] Wickens, C. D. (1992). *Engineering psychology and human performance*. HarperCollins, New York, 2nd edition.

[Wood et al., 1998] Wood, A., Aagedal, J., and Milosevic, Z. (1998). Describing virtual enterprises: the object of roles and the role of objects. In *Proc. of the OOPSLA 1998 Workshop on Virtual Enterprises*.

[Yamamoto et al., 1989] Yamamoto, Y., Kashihara, A., Kawagishi, K., and Tsukamoto, N. (1989). A tool for construction of personal database: Trias. In *The Trans. of IPSJ*, volume 30, pages 733–742.

[Yang, 2006] Yang, S. J. H. (2006). Context aware ubiquitous learning environments for peer-to-peer collaborative learning. *Journal of Educational Technology & Society*, 9(1):188–201.

[Yong, 2009] Yong, L. T. (2009). Collaborative awareness for translation groupware. In *Procs. Of the International Conference on Information and Multimedia Technology, ICIMT 2009*, pages 47–51.