Trabajo de Grado

Un Diseño Orientado a Objetos para proveer Awareness en una Hipermedia Colaborativa

---

María Verónica Iturrioz
María Guillermina López

Director: Lic. Luis Mariano Bibbó

Agradecimientos

Agradecemos a Luisma que nos guió para que podamos lograr este trabajo de grado, a la gente del LIFIA por su colaboración siempre que la necesitamos, especialmente a Javier Bazzoco.

Gracias a nuestras familias por acompañarnos y apoyarnos en todo momento.

Abstract

En este trabajo de grado presentamos un modelo orientado a objetos de awareness para ser aplicado en aquellas hipermedias que tengan características de colaboración y trabajo en grupo.

Como punto de partida exploramos los principales conceptos involucrados en el modelo: hipermedia colaborativa y awareness; además de justificar la necesidad del awareness.

En el capítulo 2 se brindan los aspectos teóricos que en conjunto definen una hipermedia colaborativa. Estos aspectos son hipermedia, groupware (aplicaciones colaborativas), comunicación, coordinación y colaboración. Uniendo estos conceptos llegamos a la definición de hipermedia colaborativa. Finalmente exponemos una metodología de diseño orientada a objetos llamada OOHDM y una extensión de ésta llamada CHDM que le agrega capacidades colaborativas.

En el siguiente capítulo se resalta la importancia y necesidad del awareness en las aplicaciones colaborativas. Brindamos una definición, introducimos las nociones de awareness, presentamos mecanismos de recolección y provisión de información de awareness, y por último una tabla con los elementos de awareness que podrían ayudar al desarrollador para determinar qué información de awareness puede ser útil para los usuarios.

La principal contribución de este trabajo de grado se encuentra expuesta en los capítulos 4 y 5, aquí presentamos y discutimos un diseño de awareness. Brindamos un modelo orientado a objetos ampliando las características de awareness de CHDM.

En el capítulo 4 proponemos como una posible implementación del awareness en una hipermedia colaborativa un framework que denominamos ACH (Awareness in Collaborative Hypermedia).

Utilizamos diagramas de interacción para mostrar la comunicación entre las clases definidas y en diferentes escenarios colaborativos. Concluimos el capítulo mencionando ventajas y desventajas del modelo y del framework y proveemos un diagrama que muestra la relación entre los distintos modelos de una hipermedia colaborativa (incluyendo el modelo de awareness) y el framework.

En el capítulo 5 para mostrar la utilidad del diseño propuesto implementamos un prototipo que llamamos InCoWeb que es una hipermedia colaborativa para la web de la facultad de informática, cuyo principal propósito es mostrar el awareness. Se instancian los modelos conceptual de OOHDM, colaborativo de CHDM y de awareness presentado en este trabajo.

Por último mencionamos y explicamos brevemente las tecnologías utilizadas para la construcción del prototipo.

Índice General

Capítulo 1

Introducción

1.1 Aplicaciones de Hipermedia Colaborativas

En los últimos tiempos, se han incrementado la cantidad de usuarios que utilizan aplicaciones de hipermedia; como son bibliotecas, enciclopedias y museos virtuales, chats, pizarrones compartidos, que les permiten compartir el espacio de navegación y la información.

Hasta hace no muchos años los usuarios de aplicaciones de computadora trabajaban en su estación de forma individual y aislada; lo que en ciertos casos complicaba el desarrollo natural de trabajo, cuando las actividades se realizaban en forma grupal.

No existía manera de tener conocimiento de lo que el resto del grupo de trabajo se encontraba haciendo en cada momento, no había posibilidad de intercambiar información.

Debido a estas limitaciones y mediante el uso de redes aparece una nueva generación de aplicaciones, que permiten a los usuarios comunicarse entre sí y colaborar en actividades; es decir aplicaciones que permiten trabajar en grupo, utilizando la computadora como un medio de trabajo en sí misma y también como una herramienta para la comunicación con el resto.

Los usuarios de este tipo de sistemas cada vez más, pueden realizar actividades colaborativas con los otros usuarios, compartiendo el espacio de navegación. Esto les permite intercambiar información, discutir algún tema o diseñar algo en conjunto.

Los ambientes de hipermedia que brindan capacidades para colaboración son llamados *hipermedias colaborativas*.

Podemos definir a los sistemas de *hipermedias colaborativas* como:

“ Ambientes de hipermedia que permite a un conjunto de usuarios navegar y manipular el contenido de manera colaborativa.”

Las primeras aplicaciones de hipermedia fueron aplicaciones que no permitieron cambios, posiblemente como consecuencia de propiedades físicas de su medio de soporte.

El mayor crecimiento en el uso de aplicaciones colaborativas lo tuvo Internet. Como punto de partida se encuentra la WWW como repositorio común de información accesible globalmente (esto se podría considerar una forma primitiva de colaboración). Por otro lado, en Internet, se

dispone de otras formas de compartir información mediante el uso de aplicaciones de e-mail, ftp, etc.

Últimamente la web ha evolucionado brindado aplicaciones que permiten a los usuarios colaborar, además de compartir información. Ejemplos de este tipo de aplicaciones son: chat, ICQ, messenger, foros de discusión para intercambio de información, sistemas de videconferencias, aplicaciones que permitan a los usuarios comentar sobre determinado tema o libro.

Estos ejemplos demuestran el interés que hay en colaboración y cooperación alrededor de la web.

Muchos sistemas colaborativos fueron desarrollados para plataformas particulares y podían usarse sólo dentro de organizaciones particulares. La WWW por otro lado ofrece una infraestructura independiente de la plataforma accesible globalmente.

Al diseñar una hipermedia colaborativa deben considerarse aspectos colaborativos como son, la cantidad de información que se comparte, qué actividades se realizan grupalmente, modelado de usuarios y roles, coordinación en la interacción entre usuarios, provisión de medios de comunicación y *awareness.*

Este trabajo de grado se centra en uno de los aspectos anteriormente mencionados, el *awareness.*

1.2 Awareness

Podemos definir al a*wareness* como el conocimiento de las acciones del resto de los usuarios con el ambiente de hipermedia para permitir que dichos usuarios puedan colaborar eficientemente.

La evolución de las tecnologías, el auge de la utilización de Internet en los últimos años, dio lugar al crecimiento de las aplicaciones de hipermedia.

Como mencionamos anteriormente el éxito de las aplicaciones de mensajería instantánea como ICQ y aplicaciones de chat, como también aplicaciones cooperativas soportadas por computadoras, han demostrado el interés que hay en colaboración y cooperación alrededor de la web.

Para lograr que una aplicación de hipermedia colaborativa en la web sea exitosa, al estar el grupo de trabajo distribuido, surge la necesidad de que cada persona del grupo requiera interactuar con el resto de las personas de una manera distinta a la que existe si dichas personas se encuentran en el mismo lugar.

Al estar distribuidos, no existen gestos, miradas, percepciones, el campo visual está reducido al área de la pantalla. Es necesario entonces que la aplicación brinde ciertos aspectos que ayuden a las personas para poder llevar a cabo la tarea en forma eficiente y consciente de que es parte de un grupo.

Necesitamos que las personas que estén colaborando en una actividad sean conscientes de las interacciones del grupo para poder colaborar eficientemente, esto es contar con información awareness.

1.3 Objetivo

El objetivo de esta tesis es brindar un diseño orientado a objetos que permita modelar el awareness en hipermedias colaborativas. El diseño conceptual y el de implementación de hipermedias colaborativas son tareas complejas de llevar a cabo. El modelo propuesto en este trabajo se basa en OOHDM, que es una metodología de diseño que brinda asistencia en la creación de modelos para ambientes hipermediales single-user, y en CHDM, la cual es una extensión de OOHDM que brinda informalmente soporte para la creación de modelos de hipermedias colaborativas. En esta tesis realizamos una extensión y nos centramos en el modelo del awareness de hipermedias colaborativas.

Brindaremos junto al modelo, un framework de awareness el cual facilitará y conducirá la implementación de aquellas hipermedias a las que se les desee agregar características de awareness. Además, implementaremos un prototipo utilizando dicho framework que demostrará la utilidad y eficacia del mismo.

Capítulo 2

Hipermedia Colaborativa

Las aplicaciones de hipermedia se caracterizan por la representación de información desestructurada que permite relacionar cualquier pieza de información con cualquier otra. Están formadas por una red de nodos y links, por la cual, los usuarios pueden navegar libremente sin un orden preestablecido.

Una de las principales plataformas de implementación para aplicaciones de hipermedia es la WWW, que también es un punto de partida para compartir conocimiento. La web tiene muchas ventajas para ser usada como plataforma en los sistemas colaborativos, los cuales requieren que las personas interactúen, compartan información y colaboren en trabajos comunes.

2.1 Hipermedia

La manera más fácil de definir hipertexto es contrastándolo con los libros tradicionales de texto, los cuales son secuenciales, es decir, existe una única secuencia lineal que define el orden en el cual el texto es leído, primero la página 1, luego la 2, y así siguiendo.

El hipertexto es no secuencial, no hay un único orden que determine la secuencia en el cual el texto es leído. El hipertexto presenta diferentes opciones a los lectores, cada uno determina qué leerá luego en el momento en el que está leyendo.

Hipertexto consiste de piezas de texto u otra información entrelazadas. Cada unidad de información se denomina nodo, cada uno de ellos puede tener punteros a otras unidades, estos punteros son llamados links.

La estructura entera del hipertexto forma una red de nodos y links.

Los lectores se mueven alrededor de esta red, y a esta actividad se la llama navegación, para enfatizar que los usuarios pueden determinar el orden en el cual visitar los nodos. Los links se asocian frecuentemente a partes específicas de los nodos que ellos conectan, en lugar de a los nodos como un todo.

La definición tradicional del término hipertexto implica que este es un sistema para tratar texto. Dado que muchos sistemas incluyen otros tipos de medios, se utiliza el término *hipermedia* para contemplar aspectos multimediales en el sistema, es decir, una *hipermedia* es un hipertexto multimedial.

No todas las aplicaciones son de naturaleza hipermedial. Para determinar cuándo una aplicación se adapta a las características de los

sistemas de hipermedia, Sheiderman [Shn 89] propuso lo que él llamó *"las tres reglas de oro del hipertexto":*

- Gran contenido de información organizado en numerosos fragmentos
- Los fragmentos relacionados unos con otros
- El usuario sólo necesita una pequeña fracción de información en un momento determinado

Cualquier sistema de información podría incrementar su usabilidad y utilidad usando características de las aplicaciones de hipermedia. Entre las ventajas de un sistema hipermedial es que permiten organizar estructuras flexibles para representar la información que pueden ampliarse fácilmente y son de fácil mantenimiento.

2.1.1 Usabilidad de la Hipermedia

La usabilidad está asociada con cinco atributos [Nie95]

- Fácil de aprender: el usuario puede rápidamente entender los comandos básicos y opciones de navegación y usarlos para ubicar la información que necesita
- Eficiente para usar: cuando un usuario ha aprendido el sistema, es posible lograr un alto nivel de productividad
- Fácil de recordar: luego de un período de no usar la hipermedia los usuarios no tienen problema en recordar cómo usarlo y navegar en él
- Pocos errores: los usuarios no cometen muchos errores mientras usan el sistema. Si navegan a un nodo equivocadamente es fácil para ellos retornar a la ubicación anterior
- Placentero de usar: los usuarios prefieren usar sistemas de hipertexto a soluciones en papel o sistemas no hipertextuales

2.1.2 Hipermedia en Internet

Internet provee muchos servicios, algunos de los más populares son el correo electrónico, transferencia de archivos, y la habilidad de correr software sobre computadoras ajenas y ver los resultados sobre la pantalla de nuestra propia computadora.

Internet puede ser usada también para hipertexto, y ese uso ha crecido mucho desde 1992. El uso de hipertexto sobre Internet ha crecido mucho más rápido que el resto de los servicios como ser FTP.

Existen varias razones de la popularidad del hipertexto sobre Internet. Las interfases de usuario tradicionales para Internet eran muy difíciles de usar y requerían que los usuarios entiendan un lenguaje de comandos con abreviaciones complejas y opciones confusas. El enfoque alternativo de tener en la pantalla de la computadora las opciones y permitir al usuario cliquear sobre imágenes o descripciones de la información en lenguaje natural es mucho más intuitivo.

El sistema más ampliamente usado para acceso de hipertexto sobre Internet es la World Wide Web. La Web usa una arquitectura cliente/servidor para hipertexto distribuido que puede ser accedida sobre Internet.

La emergencia de la WWW ha dado una nueva generación de sistemas de información; los que combinan la navegación a través de espacios de información heterogéneos con operaciones de consulta o que afectan esa información.

La WWW provee una arquitectura simple cliente/servidor , y lo más importante, desde el punto de vista del diseño de aplicaciones, introduce el paradigma de hipertexto (o hipermedia).

Esencialmente la Web sigue la arquitectura de 3 niveles recomendada para sistemas de hipertexto. El nivel más bajo, el de base de datos, consiste de Internet y de aquellas computadoras alrededor del mundo que deseen brindar materiales a otras computadoras sobre la Web. Esas computadoras actúan como servidores y en principio al usuario no le importa saber dónde están ubicadas, qué tipo de hardware o software usan o qué mecanismos de almacenamiento interno usan. Todos los servidores proveen sus datos al software cliente en un formato estandarizado llamado HTML (hypertext markup language) a través de un protocolo de comunicación estándar llamado HTTP (hypertext transfer protocol). La combinación de HTML y HTTP constituye la máquina abstracta de hipertexto y es el único punto en el cual el cliente y el servidor necesitan acordar (segundo nivel). El nivel de presentación del modelo está manejado por el browser cliente corriendo sobre la máquina del usuario.

Durante muchos años la WWW fue utilizada únicamente como un medio de difusión de información que permitía a personas distribuidas por todo el mundo compartir ideas y conocimiento. Desde ese punto de vista se puede decir que brindaba un mínimo mecanismo de colaboración (permitía a las personas compartir información).

Aplicaciones como la Web y los browsers de usuario abrieron grandes oportunidades para la colaboración entre diferentes tipos de usuarios de cualquier disciplina en el mundo.

La ausencia de un propietario sobre Internet hace que sea un sistema abierto, donde los usuarios de todo el mundo se pueden comunicar e interactuar con cualquier otro sin la necesidad de tener aplicaciones específicas.

2.2 Aplicaciones Colaborativas - Groupware

La sociedad adquiere muchos de sus caracteres de la forma en la cual las personas interactúan. A medida que las tecnologías de computadoras y otras formas de comunicación electrónica continúen evolucionando y convergiendo, la gente continuará interactuando en nuevas y diferentes formas.

El estudio de sistemas que integran actividades de procesamiento de información y de comunicación es parte de un campo multidisciplinario: Computer-Supported Cooperative Work (CSCW). CSCW se ocupa de estudiar la forma en la que los grupos de personas trabajan y analiza cómo la tecnología (especialmente computadoras) puede ayudar a ellos en sus trabajos. Examina el diseño, adopción y uso de groupware.

Muchas veces se utiliza el término Groupware como sinónimo de CSCW.

El objetivo de groupware es asistir a los grupos en la comunicación, en la colaboración, y en la coordinación de sus actividades.

Definimos *groupware* como:

Sistemas basados en computadoras que soportan grupos de personas involucradas en una tarea común (u objetivo) y que proveen una interfase a un ambiente compartido.

Las nociones de una *tarea común* y *ambiente compartido* son cruciales en esta definición. Esta excluye a los sistemas multiusuarios, cuyos usuarios no pueden compartir una tarea común.

El groupware que soporta actividad simultánea es llamado *real-time groupware*; por otro lado, aquel que no soporta actividad simultánea es llamado *non real-time groupware*

Groupware es la clase de aplicaciones, para grupos u organizaciones, que surge de la unión de computadoras, de grandes bases de información, y de tecnologías de comunicación (esas aplicaciones pueden o no soportar cooperación). Groupware es una tecnología diseñada para facilitar el trabajo en grupo. Se puede usar para comunicar, cooperar, coordinar, resolver problemas, competir o negociar.

El diseño de groupware involucra:

- Entender a los grupos y cómo la gente se comporta en grupos
- Entender la tecnología de red y cómo los aspectos de esta tecnología afectan la experiencia del usuario
- Todos los aspectos del diseño de la interfase de usuario
- Facilidad de uso

- La sensibilidad y la confiabilidad se vuelven puntos más importantes
- Entender el grado de homogeneidad de los usuarios
- Entender los posibles roles que la gente juega en un trabajo colaborativo

Muchos sistemas de software sólo soportan la interacción entre el sistema y un usuario. Cuando preparamos un documento, consultamos una base de datos, o jugamos un video-game, el usuario interactúa solamente con la computadora. Aún sistemas diseñados para aplicaciones multiusuario proveen soporte mínimo para la interacción entre usuarios.

Ese tipo de soporte es claramente necesario, debido a que una porción significante de las actividades de una persona ocurren en grupos, más que en un contexto individual.

Para lograr el soporte para interacción en grupo debemos tener en cuenta tres áreas claves: *comunicación*, *colaboración*, y *coordinación*.

2.2.1 Comunicación, colaboración y coordinación

La *comunicación* mediada por computadoras, tales como el correo electrónico, no está totalmente integrada con otras formas de comunicación. El mundo asincrónico y basado en texto del correo electrónico y pizarras compartidas existe separadamente del mundo sincrónico del teléfono o conversaciones "cara a cara". Si bien existen programas de correo con voz, etc., existen aún agujeros entre los mundos sincrónico y asincrónico. La integración de las telecomunicaciones y de las tecnologías de procesamiento ayudará a disminuir tales agujeros.

Una efectiva *colaboración* requiere que la gente comparta información. Existen muchos sistemas de información (en particular los de bases de datos) en los cuales se aísla a los usuarios. Lo que se necesita para una buena colaboración son ambientes compartidos que ofrezcan un contexto de grupo actualizado y notificaciones explícitas de las acciones de cada usuario.

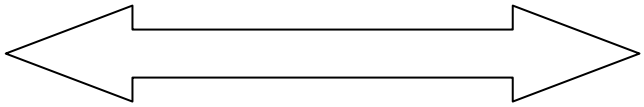
La efectividad de la comunicación y colaboración puede ser incrementada si las actividades del grupo están *coordinadas*. Sin coordinación, por ejemplo, un equipo de programadores o escritores podrían entrar en conflicto o repetir ciertas acciones realizadas por otros.

2.2.2 Espectro de groupware - Conceptos

No existe una línea rígida que divida a aquellos sistemas que son considerados de groupware de aquellos que no lo son. Los sistemas que soportan tareas comunes y ambientes compartidos varían ampliamente, es apropiado pensar en un espectro de groupware con diferentes sistemas

en diferentes puntos del espectro. El espectro tiene dos dimensiones: la de tarea compartida y la de ambiente compartido.

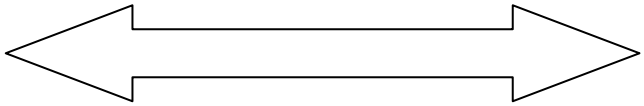
Dimensión de Tarea Común



Low
Timesharing System

High
Software Review System

Dimensión de Ambiente Compartido



Low
Electronic Mail System

High
Electronic Classroom System

Los sistemas de groupware pueden ser concebidos para ayudar a grupos en los cuales las personas se encuentren en un mismo lugar reunidos, o para grupos en los cuales sus integrantes estén distribuidos sobre varias ubicaciones.

Por otro lado, un sistema de groupware puede ser concebido para reforzar la comunicación y la colaboración dentro de una interacción en tiempo real (sincrónica), o de una interacción que no se lleva a cabo en tiempo real (asincrónica).

Esas consideraciones de tiempo y espacio sugieren cuatro categorías de sistemas de groupware representados en la siguiente tabla:

	Mismo Tiempo	**Diferente Tiempo**
Mismo Lugar	*Interacción cara a cara*	*Interacción Asincrónica*
Diferente Lugar	*Interacción distribuida Sincrónica*	*Interacción distribuida Asincrónica*

A continuación mencionaremos algunos ejemplos de sistemas que se ubican en los diferentes cuadrantes de la tabla anterior.

La tecnología *meeting room* se ubica dentro del cuadrante izquierdo superior, es decir mismo lugar - mismo tiempo: este tipo de sistemas consiste de una habitación con una gran pantalla común conectada a un

conjunto de computadoras interconectadas. Estos sistemas han surgido con el objetivo de mejorar los encuentros cara a cara y sirven para procesos como *brainstorming*, organización y evaluación de ideas. Entre los beneficios de tales sistemas se pueden mencionar:

⇒ Tipeo paralelo

⇒ Entrega rápida a todos los usuarios

⇒ Acumulación de ideas; las ideas propuestas son escritas en paralelo sin esperar un turno para proponer una idea, permitiendo que sean evaluadas por cada usuario en el momento adecuado

⇒ Anonimato opcional; el cual puede aliviar los factores de inhibición sociales

⇒ Transferencia de información basada en computadoras dentro y fuera del encuentro surgido como resultado del meeting

La tecnología de *shared whiteboard* o pizarrón compartido se encuentra dentro de la celda inferior izquierda, este tipo de sistemas fueron diseñados para dar soporte a encuentros donde se hacen discusiones acerca de diseño por ejemplo. En este tipo de encuentros las personas dibujan sus diseños y apuntan ítems particulares y relaciones. Cada persona del encuentro puede referirse a los dibujos y proponer modificaciones alterándolo. Los objetos dibujados en el pizarrón compartido están visibles a todos los usuarios.

En el cuadrante superior derecho podemos ubicar a la tecnología de *bulletin boards* o *news groups* , este tipo de sistemas son una variante a los sistemas de e-mail. En lugar de enviar un mensaje vía una computadora a una o más personas (como es el caso del mail), estos sistemas permiten a los usuarios enviar un mensaje a un "lugar" identificado unívocamente en el ciber espacio utilizado para discutir acerca de un tema en particular.

Por último el sistema de correo electrónico en la celda inferior derecha, es el ejemplo más familiar de groupware, permite intercambiar mensajes textuales en forma asincrónica. Otro ejemplo de sistemas que se pueden ubicar en esta celda son los sistemas de *group schedulling*. Programar un encuentro con un grupo de personas es una de las tareas de grupo que podría beneficiarse con el soporte de computadoras. Encontrar tiempo libre en varias agendas personales es una tarea que puede ser hecha más eficientemente mediante un sistema de groupware. Hay ejemplos de estos sistemas que utilizan un mecanismo de voto para organizar el plan de encuentro. Sistemas más nuevos involucran el intercambio de una serie de mails que son invitaciones, respuestas de confirmación y negaciones.

Un sistema de groupware podría abarcar todos los cuadrantes. Por ejemplo, sería útil tener la misma funcionalidad base y el mismo *look and feel* de la interfase de usuario (a) mientras estoy utilizando una computadora para editar un documento en tiempo real con un grupo (mismo tiempo / mismo lugar o mismo tiempo / diferente lugar) y (b) mientras estoy solo editando en mi oficina (diferente tiempo).

Existen otras dimensiones que se pueden tener en cuenta en la matriz, tales como el tamaño del grupo.

Conceptos de Groupware

Mencionaremos brevemente conceptos utilizados al trabajar con groupware para formar un vocabulario común y que pueda ser aplicado y entendido en el resto de la tesis.

Contexto compartido: un contexto compartido es un conjunto de objetos donde los objetos y las acciones ejecutadas sobre los objetos son visibles a un conjunto de usuarios. Por ejemplo en un sistema de pizarrón compartido un contexto compartido sería el pizarrón.

Ventana de grupo: una ventana de grupo es una colección de ventanas cuyas instancias aparecen sobre diferentes displays. Las instancias están conectadas. Por ejemplo, dibujar un círculo en una de las instancias implica que aparezca un círculo en el resto de las instancias.

Telepuntero: un telepuntero es un cursor que aparece sobre más de un display y que puede ser movido por más de un usuario. Cuando el telepuntero es movido sobre un display, este se mueve sobre todos los displays.

Vista: una vista es una representación visual o multimedial de alguna porción de un contexto compartido. Diferentes vistas pueden contener la misma información pero ser diferentes en sus representaciones, o pueden usar la misma representación pero referirse a diferentes porciones del contexto compartido. Por ejemplo un arreglo de números puede ser mostrado como una tabla o como un grafo.

Interacción sincrónica y asincrónica: en una interacción sincrónica las personas interactúan en tiempo real. Una interacción asincrónica es aquella en la cual las personas interactúan sobre un período extendido de tiempo y no requiere que las personas estén activas en el mismo momento, tal como en la correspondencia postal.

Sesión: una sesión es un período de interacción sincrónica soportada por un sistema de groupware.

Rol: un rol es un conjunto de privilegios y responsabilidades atribuidos a una persona, o a veces a un módulo del sistema. Los roles pueden ser atribuidos formal o informalmente.

2.2.3 Ventajas y desventajas

Desde la perspectiva del usuario, las sesiones distribuidas son experiencias muy diferentes de las sesiones "cara a cara". A continuación se listan algunas ventajas y desventajas de las sesiones distribuidas.

Entre las ventajas podemos mencionar:

- ***Incrementa el acceso a la información:*** los participantes de una sesión distribuida tienen acceso a sus archivos, residiendo cada uno en su oficina, lo que les permite fácil acceso a información importante que de otra manera podría no estar disponible durante una sesión. Esto brinda a los usuarios muchas ventajas en cuanto a comodidad y familiaridad al mantenerse en sus oficinas.

- ***Estimula el trabajo paralelo dentro del grupo:*** frecuentemente las personas se dividen en subgrupos para trabajar en diferentes partes de una tarea utilizando un protocolo social y vistas compartidas. Luego los trabajos individuales son mezclados con el resto de los trabajos del grupo con lo cual se tiene una ganancia en tiempo.
  Es común para usuarios en un grupo distribuido ausentarse por un momento (para trabajar sobre otra pantalla, tomar un café, etc.) y luego reanudar el trabajo grupal. Esto no es socialmente aceptable en un trabajo grupal "cara a cara", pero es aceptado en sesiones distribuidas. Además el trabajo grupal permite reunir múltiples perspectivas y habilidades, formar grupos con intereses comunes y facilita muchas resoluciones de problemas.

Entre las desventajas mencionamos:

- ***Se generan discusiones incomprensibles:*** las sesiones distribuidas tienen un patrón de comunicación marcadamente diferente a las sesiones "cara a cara". La comunicación no es full-duplex, sólo la voz de una persona es transmitida por vez, consecuentemente las personas tienden a querer ganar su turno y pueden llegar a ser descorteses o no cooperativos.

- ***Requiere más concentración para el trabajo en grupo:*** los usuarios comentan que en general las sesiones "cara a cara" parecen mas cortas, que se logra más en menos tiempo, y que son frecuentemente más alegres. En cambio las sesiones distribuidas requieren mayor concentración y son más cansadoras. Debido a que las discusiones son más complicadas cuando los miembros del grupo están distribuidos, los usuarios hacen un esfuerzo mayor para lograr un feed-back.

- ***Reduce la interacción social:*** las sesiones distribuidas tienden a ser más serias, los usuarios ponen el foco de atención sobre la

tarea específica y en general no surgen temas extra laborales. Esto genera una ganancia en eficiencia y tiempo, pero reduce el intercambio social. Muchas de las sesiones "cara a cara" parecieran ser más intensas, con interacciones más ricas. Si bien raramente los miembros del grupo se miran directamente entre ellos, el hecho de estar en la misma habitación incrementa el awareness de las actividades de los otros miembros.

2.2.4 Consideraciones de diseño

Esta sección se centra sobre la investigación en groupware, describiendo los problemas a los que se siguen enfrentando los diseñadores y desarrolladores.

a. Interfases de grupo

Las interfases de grupo difieren de las de único usuario en que representan actividad grupal y están controladas por múltiples usuarios. Las interfases de grupo introducen problemas de diseño, por ejemplo, cómo manejar la complejidad producida por la actividad de múltiples usuarios. Otro tema a considerar es qué técnicas y conceptos de las interfases de único usuario son útiles en las de grupo.

- ***"What You See Is What I See " - WYSIWIS:*** es un enfoque para diseñar interfases de grupo en el cual se garantiza que el contexto compartido aparecerá igual para todos los participantes. La principal ventaja es el fuerte sentido de contexto compartido y la simple implementación. Su principal desventaja es que puede ser inflexible.
  Los usuarios pueden querer control independiente de elementos como tamaño y ubicación de las pantallas, y pueden requerir cierta información a medida dentro de la ventana. Existe un enfoque WYSIWIS relajado el cual permite que cada usuario determine ciertos aspectos visuales sobre sus ventanas.

- ***Atención y distracción en el trabajo grupal:*** una buena interfase de grupo debería representar toda la actividad del grupo y al mismo tiempo no generar la distracción del usuario. Por ejemplo, cuando un usuario crea o hace scroll en una ventana de grupo, abre o cierra una ventana de grupo, o modifica un objeto sobre el que un usuario está trabajando/viendo, los otros usuarios pueden llegar a ser distraídos de su trabajo. Este es un punto en el que las interfases multiusuarios difieren de las de único usuario, en las cuales los usuarios tienen un contexto mental para interpretar los cambios en las pantallas que resultan de sus propias acciones. En cambio, con las interfases de grupo, los usuarios en general no son tan conscientes de los contextos o actividades de los otros como para poder interpretar fácilmente los cambios en la pantalla. Se requieren formas para proveer

ciertas pistas contextuales sobre las actividades del grupo.

- ***Dinámicas del grupo:*** las interfases de grupo deben ser consistentes con los patrones que usa el grupo para trabajar.

- ***Manejo del espacio en la pantalla:*** el espacio en la pantalla es un recurso limitado en aplicaciones de único usuario, pero es un problema mayor en interfases de grupo en las cuales cada usuario puede crear ventanas que aparecen sobre las ventanas de los usuarios. Por esta razón son necesarias técnicas para administrar la proliferación de ventanas.

b. Procesos de Grupo

Aquellas tareas que requieren la participación de un conjunto de usuarios son llamados procesos de grupo. Incrementan el paralelismo pero requieren coordinación, la cual puede sobrecargar al grupo y bajar su efectividad.

Entre los procesos de grupo podemos mencionar a los protocolos de grupo. Estos son estrategias que definen la manera de interactuar. Los protocolos pueden ser tecnológicos o sociales. Los tecnológicos son aquellos en los que la manera de interactuar está definida en el hardware o en el software; por ejemplo mecanismos de *floor control* en sistemas de conferencia en los que el sistema puede procesar un requerimiento de entrada al sistema por vez, imponiendo a los participantes un proceso de grupo mediante turnos. Por el otro lado el control de proceso de grupo puede ser dejado para que lo definan los usuarios según políticas sociales, por ejemplo si en el grupo existe un director, entonces que éste defina la manera de interacción.

c. Control de concurrencia

Los sistemas de groupware necesitan control de concurrencia para resolver los conflictos entre las operaciones simultáneas de los participantes. Los diseñadores de groupware deben tener en cuenta ciertos puntos relacionados a la concurrencia: interfase de grupo, tiempo de respuesta y notificación, replicación de datos, robustez, distribución.

d. Group Awareness

La provisión de group awareness brinda a cada usuario la información acerca de las tareas que han realizado el resto de los usuarios del grupo, de lo que están haciendo en este momento, y de lo que harán luego. Para proveer tal información el modelo debe capturar las relaciones entre los usuarios y las herramientas y artefactos que utilizan.

Hay que proveer una interfase de usuario al ambiente compartido para los usuarios geográficamente distribuidos, que les permita tener

conciencia del resto de los usuarios, del ambiente y de las tareas que se encuentran desarrollando, esto es proveer awareness.

En una situación real de trabajo en grupo las personas se reúnen en un mismo ambiente en donde la interacción es cara a cara, lo que permite comunicación no sólo verbal sino también física, como gestos, miradas, percepciones; cada uno conoce ciertas cuestiones sociales como ser el rol de cada persona en el grupo de trabajo, a qué área de trabajo pertenece cada uno, y cómo comunicarse con el otro cuando sea necesario.

Como en aplicaciones colaborativas distribuidas el grupo de personas no se encuentra en el mismo ambiente físico, las personas dentro del grupo ven reducidas sus formas de colaboración y comunicación, ya que no pueden utilizar todos sus sentidos, no existen percepciones, gestos, miradas, olores, ruidos. Surge la necesidad de que la aplicación brinde ciertos aspectos que ayuden a las personas para poder llevar a cabo la tarea de forma eficiente y siendo consciente que es parte de un grupo de trabajo; que la aplicación brinde en el mayor grado posible, de manera clara y simple, los aspectos perceptivos ausentes en un ambiente virtual, pero necesarios para una colaboración exitosa.

Considerando el auge del trabajo grupal en la web, con el uso de aplicaciones como chat, ICQ, foros de discusión, etc, y el surgimiento de nuevas aplicaciones colaborativas cada vez con diferente y mayor grado de colaboración; creemos útil y necesario que el grupo de usuarios que estén colaborando en una actividad sean conscientes de las interacciones del grupo para así poder colaborar eficientemente, esto es que la aplicación provea información de awareness.

En esta tesis proponemos un modelo orientado a objetos con el objetivo de brindar asistencia en el diseño del awareness de una hipermedia colaborativa.

Al proveer un diseño orientado a objetos del awareness un diseñador que se vea enfrentado con la necesidad de proveer información de awareness en su aplicación de hipermedia colaborativa; podría reducir el esfuerzo y tiempo que conllevan todas las fases de desarrollo de una aplicación instanciando el modelo propuesto.

e. Administración de Sesión

En un sistema colaborativo cuando un usuario colabora con otro se establece una sesión. La administración de sesión es responsable del proceso de formación de sesiones colaborativas, esto es quién se une y quién abandona una sesión.

Se pueden distinguir administración de sesión lógica y técnica. La administración de sesión lógica controla el establecimiento de la colaboración, cuáles usuarios están colaborando. La administración de sesión técnica incluye temas como encontrar servers, la conexión a base de datos y protocolos de negociación.

Las sesiones de colaboración deben estar estructuradas de acuerdo a la situación de colaboración. Los miembros se podrían unir a una sesión simplemente subscribiéndose, o podemos tener una administración de

sesión mas restringida en el que sólo se puedan unir aquellos usuarios que reciban una invitación. Otra posibilidad podría ser que los nuevos usuarios pidan permiso al resto del grupo para unirse a la sesión, lo cual puede resolverse votando o puede decidirlo el líder del grupo.

f. Control de acoplamiento

Como hay varios escenarios en una aplicación de groupware, no es posible decidir a priori cuándo un atributo concreto de la interfase de usuario (tal como *scrollbar*) tiene que estar acoplada o no.

Un escenario en el que todos los participantes tienen una vista común de la interfase de usuario es llamada ***WYSIWIS*** ("*what you is what I see*") ***fuertemente acoplado***. En este caso los objetos de la interfase de usuarios deben estar acoplados, y las acciones realizadas por un usuario sobre alguno de esos objetos, se verá reflejada en las interfases del resto de los usuarios.

Un escenario ***WYSIWIS débilmente acoplado*** es aquel en el que algunos de los objetos de la interfase de usuario no están acoplados, esto es, las acciones realizadas sobre esos objetos no se verán reflejadas en las interfases del resto de los usuarios.

Los miembros de una sesión pueden desear cambiar de un modo WYSIWIS fuertemente acoplado a un modo débilmente acoplado de acuerdo a la tarea corriente. Por ejemplo, si una lista de selección de la interfase de usuario está fuertemente acoplada, se puede dar una situación en el que un usuario seleccione un ítem y antes de que realice una determinada acción, llega otro usuario y le cambia la selección. En estos casos queremos tener un acoplamiento débil, por lo tanto, es necesario tener un control de acoplamiento.

Un tercer escenario sería aquel en el que cada usuario tiene una vista diferente del modelo de la aplicación, esto es ***vistas independientes.***

Si las sesiones colaborativas tienen un alto nivel de acoplamiento, todos ven los cambios provocados por los demás, por lo tanto, se requiere menos esfuerzo para brindar información de awareness. Cuando los usuarios trabajan de manera fuertemente acoplada, tienen la misma vista de la aplicación, por lo que se reduce la cantidad de información necesaria acerca de las acciones del resto de los usuarios del grupo. Cuanto más acoplada estén las interfases de los usuarios menor es el esfuerzo requerido para brindar información de awareness.

g. Floor control

El floor control es el encargado de moderar la cooperación entre los participantes durante una sesión, se encarga del cómo, cuándo y por qué los participantes interactúan en un ambiente compartido; provee los derechos de acceso sobre la aplicación. El término *floor* se refiere a "el turno para hablar".

Dependiendo de la tarea corriente del grupo, será apropiado un modo diferente de colaboración. Un modo sería permitir a todos los usuarios

realizar cualquier acción, esto es un modo de trabajo paralelo entre los usuarios; otro modo sería permitir sólo actuar a un usuario por vez.

Los diseñadores de groupware no siempre quieren dar un control completo a todos los usuarios sobre la aplicación. Puede querer restringirse ciertos derechos a determinados usuarios o grupos de usuarios. Con esto surge la noción de rol de usuario, los cuales abstraen las características comunes de grupos de usuarios, como derechos de acceso y de uso.

Desde el punto de vista del diseño orientado a objetos se identifican requerimientos adicionales como por ejemplo facilidad de uso, un alto nivel de abstracción, una solución consistente que capture todos los conceptos del dominio del problema, reusabilidad de componentes.

Conclusiones de groupware

El diseño orientado a objetos para aplicaciones de un único usuario tiene una larga historia. Existen diversas metodologías, patrones de diseño, frameworks, etc. que asisten a los desarrolladores de este tipo de aplicaciones. Sin embargo, esos conceptos no son aplicables directamente a las aplicaciones colaborativas sincrónicas donde deben considerarse aspectos adicionales. El paso de desarrollar aplicaciones de único usuario a desarrollar aplicaciones de groupware requiere no sólo compartir artefactos comunes o conectar un conjunto de interfases de usuarios distribuidos. Esos aspectos adicionales pueden ser vistos desde un punto de vista conceptual o técnico. Algunos aspectos técnicos a considerar son arquitecturas distribuidas, control de concurrencia, actualización de vistas, etc.

Como mencionamos anteriormente groupware es una tecnología diseñada para facilitar el trabajo de grupos. Esta tecnología puede ser usada para comunicar, cooperar, coordinar, resolver problemas, competir o negociar.

Para el diseño de groupware es importante entender los grupos de personas para los cuales se destinará el groupware, cómo la gente se comporta en grupo y el tipo de tareas que van a realizar. Es importante también hacer un análisis del grado de homogeneidad de los usuarios, como también los posibles roles que la gente juega en el trabajo cooperativo. Cuanto más homogéneo es el grupo de personas, más fácil será la comunicación entre ellos.

Hacer una aplicación de groupware exitosa es una tarea mucho más compleja que el diseño de una aplicación de software tradicional. Típicamente, un sistema de groupware no tendrá éxito a menos que muchos o todos los usuarios a los que está destinado el software lo adopten. Por el contrario, un software tradicional puede ser exitoso aún si sólo una fracción de los usuarios destinatarios lo usan.

2.3 Hipermedia Colaborativa

Hasta este momento vimos lo que es una hipermedia y en que consisten las aplicaciones colaborativas. Es tiempo entonces de unirlas y explicar a qué nos referimos con hipermedia colaborativa.

Los ambientes de hipermedia pueden ser usados por un conjunto de usuarios que comparten el espacio de navegación y la información del sistema. Estos usuarios podrían querer que el sistema les dé la posibilidad de realizar tareas colaborativas con el resto de los usuarios.

Estos ambientes de hipermedia que brindan capacidades para colaboración son llamados *Hipermedia Colaborativa*. Un ambiente de hipermedia colaborativa permite a un conjunto de usuarios navegar y manipular estructuras de hipermedia en una manera colaborativa.

Los ambientes de hipermedia colaborativa tienen diferencias con los ambientes de único usuario. Como dijimos anteriormente en el punto 2.2.1 para lograr interacción en grupo se necesita soporte para colaboración ,coordinación y comunicación entre usuarios.

Comparado al diseño de un ambiente de hipermedia convencional, el diseño de un ambiente de hipermedia colaborativa involucra, como mencionamos en el diseño de groupware, nuevos aspectos a tener en cuenta. [BZSS]

Este trabajo de grado se centra sólo en uno de estos aspectos, el *group-awareness* que se refiere al conocimiento que cada usuario tiene de información referente al resto de los usuarios dentro del ambiente colaborativo. El resto de los aspectos colaborativos son sólo tenidos en cuenta pero no explorados profundamente.

2.3.1 Hipermedia colaborativa e Internet

La WWW fue diseñada para acceder información, pero últimamente el interés de la comunidad de Internet ha puesto su foco sobre la interacción. Para permitir las colaboraciones e interacciones entre usuarios se requiere una infraestructura independiente de la plataforma accesible globalmente. Los usuarios de Internet se ven beneficiados cuando son capaces de “encontrarse” cuando acceden a los mismos datos pudiendo interactuar y colaborar entre ellos basados sobre intereses comunes.

Las primeras aplicaciones de hipermedia, fueron aplicaciones que no permitieron cambios, posiblemente como consecuencias de propiedades físicas de su medio de soporte.

La evolución de las tecnologías, el gran crecimiento de la Web, ha permitido nuevas aplicaciones que son constantemente modificadas, enriquecidas con nuevos servicios y nuevas características de navegación e interfase que son incorporadas de acuerdo a las políticas de marketing de las organizaciones.

Las ventajas de usar la Web como una plataforma para los sistemas colaborativos son:

- Plataforma universal
- De dominio público
- Fácil de usar
- Cumple con los estándares
- Es abierta e independiente de la plataforma, por lo tanto constituye una base colaborativa que cruza los límites organizacionales

Entre las limitaciones se encuentran:

- No hay percepción, por lo que se hace difícil mantener una memoria grupal
- El protocolo que usa es inherentemente sin estado, no se almacena ningún tipo de información entre los pedidos. El servidor no tiene idea de qué página el cliente está navegando. Si algo cambia en la aplicación no hay manera de que el usuario perciba los cambios
- La Web es de naturaleza asincrónica, por lo tanto cualquier aplicación o herramienta colaborativa que requiera un grado significativo de interacción sincrónica entre los usuarios y/o aplicaciones no pueden ser soportadas por la arquitectura actual de la Web.

En los últimos tiempos se ha propuesto agregar gente a la web, asociándola con espacios virtuales, pero este enfoque no tuvo realmente éxito, debido probablemente a que muchos de los modelos requerían un browser modificado, o en el mejor de los casos un plug-in. Por ésto, en los últimos años, navegar se ha vuelto una actividad individual.

Por otro lado el éxito de aplicaciones de mensajería instantánea como ICQ y aplicaciones de chat, como también la emergencia de aplicaciones cooperativas soportadas por computadoras muestran el interés en colaboración y comunicación sincrónica alrededor de la Web y la existencia de una infraestructura estable para navegación colectiva.

Creemos que la tendencia es usar la Web como una plataforma para trabajos cooperativos en los cuales los requerimientos de interacción se incrementen.

2.4 Diseño de Hipermedia Colaborativa

En este apartado describiremos una metodología de diseño que brinda asistencia en la creación de modelos para ambientes hipermediales single-user llamada OOHDM. Dado que dicha metodología no considera

aspectos colaborativos surge CHDM, la cual es una extensión de OOHDM que brinda informalmente soporte para la creación de modelos de hipermedias colaborativas.

2.4.1 OOHDM - Object-Oriented Hypermedia Design Method [Ros96]

La emergencia de la WWW ha dado una nueva generación de sistemas de información que combinan la navegación a través de espacios de información heterogéneos con operaciones de consulta o que afectan la información.

Las tradicionales metodologías de ingeniería de software no contienen abstracciones útiles capaces de facilitar la tarea de especificar aplicaciones que incluyan la metáfora de hipertexto; no incluyen ninguna noción de link y poco se dijo sobre cómo incorporar hipertexto dentro de una interfase.

Dado que el tamaño, la complejidad y el número de aplicaciones crece, se necesita un enfoque que ayude a abordar estos temas.

Las metodologías de diseño modernas tienden a emplear, o bien métodos orientados a objetos que no proveen primitivas para especificar la navegación, o bien enfoques de diseño de hipermedia que enfatizan aspectos estructurales pero ignoran el comportamiento computacional.

En el dominio de hipermedia hay requerimientos en conflicto que deben ser satisfechos en un framework unificado. Por un lado en la aplicación final la navegación y el comportamiento funcional deben ser integrados. Por otro lado durante el proceso de diseño se debe poder desacoplar las decisiones de diseño relacionadas con la estructura navegacional de la aplicación de aquellas relacionadas con el modelo del dominio.

OOHDM provee un método de alto nivel de abstracción y mecanismos de composición para resolver muchos de los problemas previamente mencionados.

De acuerdo a OOHDM, el desarrollo de aplicaciones de hipermedia ocurre como un proceso de 4 actividades: Diseño Conceptual, Diseño Navegacional, Diseño de Interfase Abstracta e Implementación, que son ejecutados en una mezcla de estilos de desarrollo incremental e iterativo, en cada paso un modelo es construido o enriquecido.

Explicaremos brevemente de qué trata cada etapa de la metodología OOHDM

Diseño Conceptual

Se comienza con la elaboración de un modelo del dominio de la aplicación. Esto es hecho durante la fase del Diseño Conceptual usando principios bien conocidos de modelado orientado a objetos, aumentado con algunas primitivas como perspectivas de atributos y subsistemas.

Las clases conceptuales pueden construirse utilizando jerarquías de agregación y de generalización/especialización. El resultado de esta etapa

es un esquema de clases y objetos construido a partir de subsistemas, clases y relaciones.

Lo más interesante de este paso es capturar las semánticas del dominio lo más neutralmente posible, con poco interés sobre los tipos de usuarios y de tareas. Si la aplicación involucra computaciones sobre objetos, como sistemas de información basados en Web, el modelo conceptual se desarrollará dentro de un modelo de objetos que será implementado en el ambiente origen (server WWW).

Diseño Navegacional

Una de las características que distingue a los sistemas de hipermedia es la noción de navegación.

En OOHDM una aplicación es vista como una visión navegacional del modelo conceptual. La visión es construida en la fase del Diseño Navegacional. Muchas vistas pueden ser construidas a partir del mismo modelo conceptual, permitiendo la construcción de diferentes modelos de acuerdo a los diferentes perfiles de los usuarios.

El diseño de la navegación se expresa en 2 esquemas, el esquema de Clases Navegacionales y el esquema de Contextos Navegacionales. Hay un conjunto predefinido de clases navegacionales: nodos, links y estructuras de acceso. Las semánticas de los nodos y links son las usuales, previamente descriptas, y las estructuras de acceso, como índices y tours guiados, representan posibles formas de acceder a los nodos. Se permite que un nodo sea definido combinando atributos de diferentes clases relacionadas en el esquema conceptual.

Un contexto navegacional es un conjunto de nodos, links, clases contextuales y otros contextos navegacionales (anidados). Los contextos navegacionales sirven para organizar el espacio navegacional en conjuntos consistentes que pueden ser atravesados siguiendo un orden particular, deben ser definidos para ayudar al usuario a ejecutar su tarea de interés.

Diseño de Interfase Abstracta

Una vez que la estructura navegacional de la aplicación ha sido definida, se especifican sus aspectos de interfase. Esto es, la forma en la cual aparecerán diferentes objetos navegacionales, cuáles objetos de interfase activarán la navegación y otras funcionalidades de la aplicación, y cuáles transformaciones de interfase tomarán lugar y cuándo. Es decir cómo se presentará al usuario la estructura navegacional.

La clara separación entre el diseño navegacional y el de interfase abstracta, permite construir diferentes interfases para el mismo modelo navegacional.

En OOHDM se usan Abstract Data View para describir la interfase. Los ADV's son objetos en el sentido que poseen un estado y una interfase de mensajes. A su vez los ADV's representan Abstract Data Objects (ADO's) de la aplicación. Un ADV está relacionado con un correspondiente objeto de aplicación el cual actúa como servidor de

comportamiento para aquellas operaciones no especificadas en la interfase.

En OOHDM los elementos navegacionales funcionan como ADO's a los que se les definirán ADV's para especificar su apariencia frente al usuario.

Implementación

La última etapa es la de implementación , en donde básicamente, hay que traducir el Modelo Navegacional y el de Interfase Abstracta a un ambiente de implementación.

En la siguiente tabla se resumen las diferentes etapas de la metodología OOHDM destacando sus conceptos más importantes.

Paso	Productos	Mecanismos	Puntos de diseño
Análisis de dominio	Clases, subsistemas, relaciones, perspectivas de atributos.	Clasificación, composición, generalización, espec.	Modelando las semánticas del dominio de la aplicación.
Diseño Navegacional	Nodos, links, estructuras de acceso, contextos navegacionales, estructuras navegacionales.	Mapeo entre objetos conceptuales y navegacionales.	Énfasis sobre aspectos cognitivos
Diseño de Interfase Abstracta	Objetos de interfase abstracta,respuestas a eventos externos, transformaciones de interfaz.	Mapeo entre objetos perceptibles y de navegación	Modelando objetos perceptibles. Describe interfase para objetos navegacionales.
Implementación	Corriendo la aplicación	Lo provisto por el ambiente destino.	Performance.

Resumen de metodología OOHDM

Contribuciones de OOHDM

Según [Ros96], las principales contribuciones de OOHDM son:

- Una separación clara entre la etapa de análisis y diseño, dada la diferencia de primitivas de modelado para cada etapa.

- Una visión completa de los proyectos de aplicaciones de hipermedia teniendo en cuenta sus diferentes actividades.

- Un conjunto de formalismos orientados a objetos y mecanismos de abstracción que soportan la construcción de aplicaciones de hipermedia.

- Un conjunto de patrones de arquitectura de hipermedia para ser utilizados durante el ciclo de desarrollo.

- Un modelo simple de documentación para ayudar durante la evolución de la aplicación.

Limitaciones de OOHDM para hipermedias colaborativas

OOHDM es una metodología de diseño que asiste en la creación de modelos para ambientes hipermediales single-user, sin embargo carece de soporte para características colaborativas. No fue pensada para hipermedias colaborativas, por lo tanto carece de algunas características propias de colaboración; entre estas carencias están:

- No permite modelar a los usuarios en particular y las relaciones entre ellos.

- Modela principalmente aspectos estructurales y no dinámicos en el diseño de las estructuras navegables.

- No permite vincular herramientas de edición o colaboración a la estructura de navegación.

- Para poder tener diferentes tipos de usuarios que usen la aplicación, se debe definir un modelo de navegación para cada uno de ellos. Pero OOHDM no permite modelar a los usuarios en particular, ni los diferentes roles que pueden jugar en la aplicación, ni las relaciones entre ellos, o la posibilidad de interacción y comunicación. Al no permitir modelar a los usuarios en particular, no soporta modelar características colaborativas como el *awareness*.

- OOHDM sólo modela a las aplicaciones de hipermedia como un conjunto de nodos en los que se puede navegar , pero no contempla que dichos nodos tengan herramientas colaborativas indispensables para la colaboración.

- OOHDM modela los aspectos estructurales de las aplicaciones de hipermedia, pero en las aplicaciones de hipermedia *colaborativas* es necesario poder modelar aspectos de comportamiento, ya que además de interactuar solos con el sistema, los usuarios también interactúan entre sí.

2.4.2 CHDM – Collaborative Hypermedia Design Method **[MR01]**

En [BZSS] se propone una extensión a OOHDM para soporte de características colaborativas de una manera informal.

Un modelo se califica como informal cuando está expresado mediante lenguaje natural, figuras, tablas u otras notaciones. Por otro lado, nos referimos a modelos formales cuando la notación empleada es un formalismo, es decir posee una sintaxis y semántica precisamente definidas.

Existen lenguajes gráficos de modelado exitosos tales como UML (Unified Modeling Languaje), los cuales se basan en el uso de construcciones gráficas que transmiten un significado intuitivo y que resultan fáciles de entender y aplicar por los usuarios. Sin embargo, la falta de precisión en su semántica puede generar problemas de interpretación entre los desarrolladores, o inconsistencias entre los diferentes modelos de una misma aplicación. Por otro lado, los lenguajes formales de modelado poseen una sintaxis y semántica bien definidas, pero son menos utilizados debido a la complejidad de sus formalismos matemáticos que son difíciles de entender.

En CHDM se establece una metodología de diseño para aplicaciones de hipermedia colaborativa denominada CHDM (Collaborative Hypermedia Design Method). Dicha metodología prioriza el diseño de las aplicaciones de hipermedia a las tecnologías sobre las que se desarrollarán, y provee una base formal que permite realizar verificaciones de invariantes o reglas de buena formación en los modelos resultantes del uso de esta metodología.

La metodología CHDM es una extensión de OOHDM para soporte de características colaborativas. CHDM reusa las etapas del modelo conceptual, modelo navegacional, diseño de la interfase abstracta e implementación definidas en OOHDM y agrega una más, la etapa del modelo colaborativo. Es decir, plasma en diagramas de clases UML un diseño orientado a objetos para las etapas conceptual y navegacional definidas en OOHDM; y luego, en base a dichas etapas, define y elabora un diseño para la etapa colaborativa. Las etapas de diseño de Interfase Abstracta e Implementación definidas en OOHDM dependen fuertemente de la arquitectura y herramientas que se empleen.

Modelo colaborativo

El modelo colaborativo propuesto en CHDM tiene en cuenta ciertas características de las aplicaciones colaborativas tales como identificación y manejo de los usuarios de la aplicación y sus roles, sesiones de colaboración entre ellos junto con la manera en que interactúan y colaboran, control de concurrencia, awareness, uso de herramientas colaborativas sincrónicas y asincrónicas, etc.

La etapa del diseño colaborativo se especifica posteriormente al diseño del modelo navegacional y consiste en identificar los nodos de la

aplicación a los que se les agregará alguna de dichas características colaborativas y modelar nodos colaborativos para ellos.

Como resultado de la nueva etapa se obtiene un modelo de objetos concreto para especificar de manera estructural las características colaborativas de la aplicación, y diagramas de secuencia que permiten especificar la parte dinámica de la aplicación, es decir, el comportamiento de dichos objetos en escenarios colaborativos puntuales y en qué manera colaboran.

Un diseñador de aplicaciones de hipermedias colaborativas debe instanciar cada uno de los modelos de CHDM. Tanto los diagramas estáticos como los dinámicos surgen de la instanciación de los modelos propuestos por CHDM. Además, el diseñador puede subclasificar las clases que necesite, como así también agregar diagramas de secuencia para una mejor comprensión del modelo.

A continuación mostramos el metamodelo colaborativo definido por CHDM. Obviamos los metamodelos conceptual y navegacional debido a que son los mismos definidos en OOHDM.

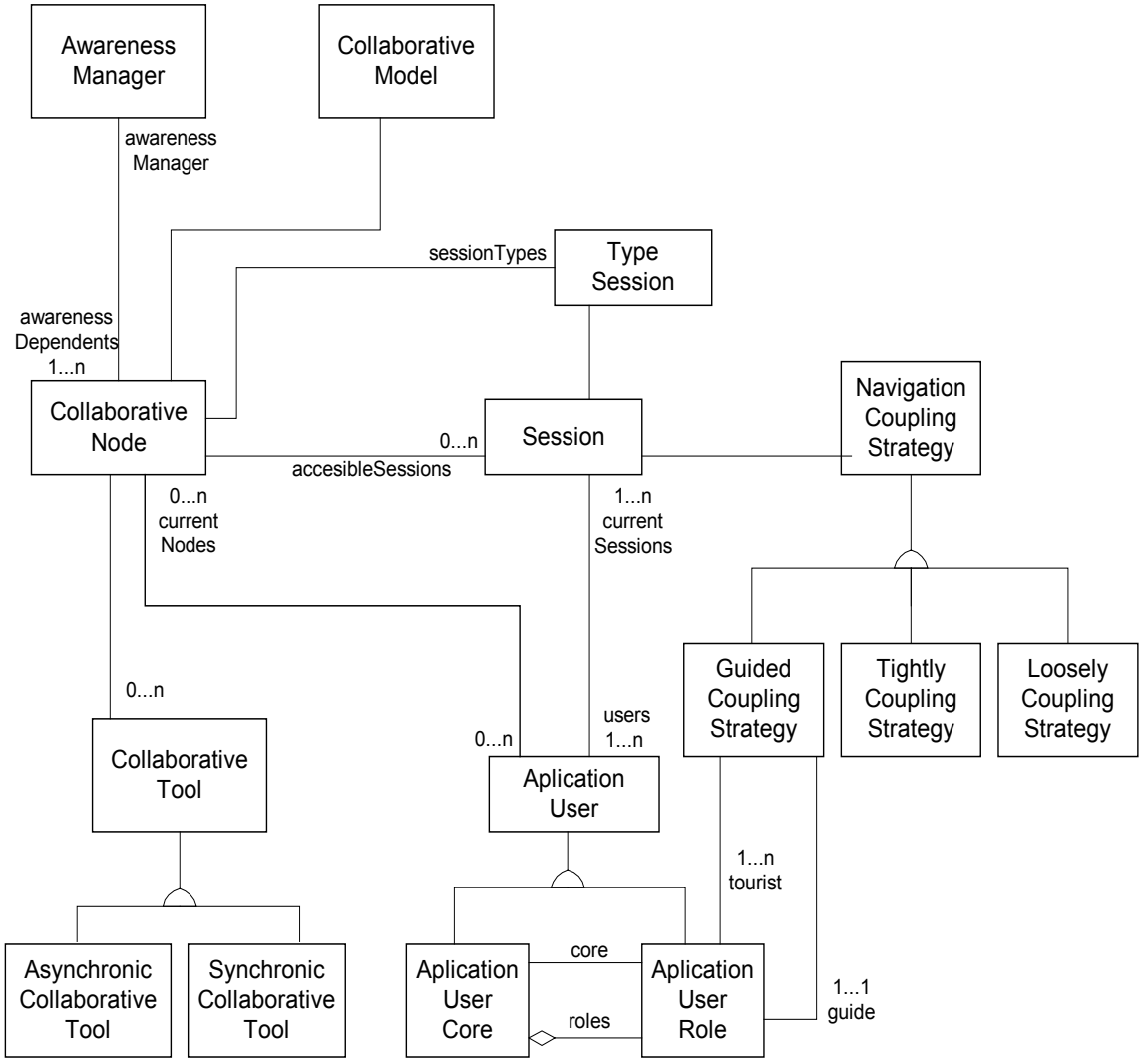


Figura 2.1

Relación entre los modelos de CHDM

En la figura 2.2 se muestran las conexiones existentes entre el modelo conceptual y el navegacional. Una de las conexiones entre dichos modelos es entre las metaclases *Class* y *Node*.

Existe una relación de dependencia: un nodo observa a varios objetos de los cuales es dependiente, y de los cuales obtiene atributos para mostrar.

La segunda conexión entre los modelos conceptual y navegacional se encuentra entre las metaclases *KnowledgeRelationship* y *ApplicationLink*. Un link de la aplicación mapea una relación existente entre objetos de la aplicación.

Luego, se muestra la relación existente entre los modelos navegacional y colaborativo, y se ve reflejada mediante las metaclases *CollaborativeNode* y *Node*. Esta relación cumple con el pattern *Decorador* [GHJV94–pag175] debido a que el nodo colaborativo agrega al nodo ciertas características colaborativas, tales como awareness, control de concurrencia, grupos de usuarios colaborando, herramientas colaborativas, etc.

Nótese que la metaclase *ApplicationLink* del modelo navegacional no tiene una representación en el modelo colaborativo, pues los links no requieren características colaborativas como los nodos.

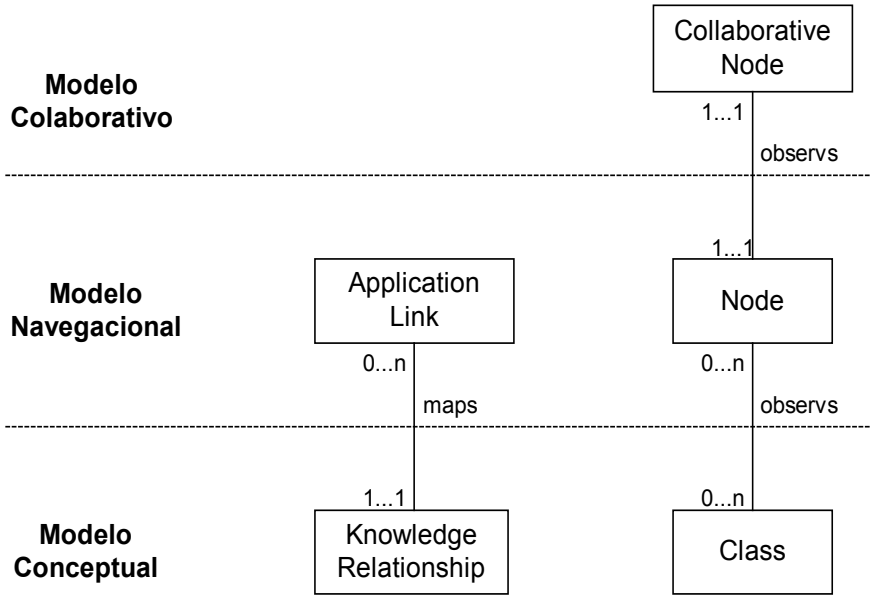


Figura 2.2

Capítulo 3

Awareness – Aspectos teóricos

La rica interacción que existe entre las personas en ambientes físicos reales permite a la gente mantener conocimiento y conciencia instantánea acerca de la presencia y comportamiento de las otras personas con el ambiente, como ser qué objetos están manipulando o el nivel de actividad. En la comunicación mediante computadoras existe una dificultad inherente al medio, ya que la visibilidad se reduce o elimina, se pierden los sentidos y percepciones, se puede tener la sensación de estar trabajando solo.

Históricamente, los sistemas y aplicaciones multiusuario trataban de aislar a los usuarios y sus trabajos de los demás, para dar la ilusión de que el sistema tenía sólo un usuario. Pero en el trabajo colaborativo, donde las personas del grupo de trabajo necesitan comunicarse e interactuar no sólo con la computadora sino entre ellos, es fundamental que no exista aislamiento para que puedan compartir información y que tengan conciencia de que están cooperando para un fin común. Necesitamos que los ambientes colaborativos – aplicaciones de groupware – provean información de *awareness* para que los usuarios obtengan dicha conciencia grupal.

3.1 Definición

El *awareness* es un concepto que surge de las aplicaciones de groupware y lo podemos definir como:

Conocimiento que cada usuario tiene en todo momento de la interacción de los otros usuarios con la aplicación.

Ese conocimiento es fundamental para poder llevar a cabo tareas colaborativas en las que se requiere comunicación y coordinación entre los usuarios, de forma similar a la manera en que se interactúa en grupos de trabajo de la vida real.

Dourish y Bellotti [DB92] definen *awareness* como "*entender las actividades de los otros, lo que provee un contexto para nuestra propia actividad*". Este contexto es usado para asegurar que las contribuciones individuales son relevantes a la actividad del grupo como un todo, y para evaluar acciones individuales con respecto a las metas y progresos del grupo. El contexto dentro del cual los miembros del grupo colaboran abarca no sólo el contenido de las contribuciones individuales, sino

también su carácter, es decir su significancia con respecto al grupo como un todo y a sus metas.

3.2 Importancia del Awareness – Características

La información que se obtiene a partir del *awareness* permite a los grupos manejar el proceso de trabajo colaborativo.

El *awareness* de las actividades individuales y de grupo es crítico para el éxito de la colaboración, para la coordinación de las actividades y para compartir la información. Sin *awareness* estas actividades se vuelven anti-sociales y se pierde la facilidad y naturaleza de la colaboración; haciéndola difícil, ineficiente y torpe, comparado con un trabajo colaborativo en un ambiente físico real. [Moody00]

El *awareness* es fundamental para actividades colaborativas y sociales. Si uno observa a los grupos de personas en situaciones de la vida real, tener conocimiento del resto de las personas del grupo es el primer paso hacia cualquier clase de interacción. Si uno no es consciente de la presencia de las otras personas del grupo, la interacción se vuelve más compleja y puede llegar a no ser posible una interacción exitosa.

El *awareness* tiene cuatro características básicas:

- ✓ Es el conocimiento acerca del estado de un ambiente particular.
- ✓ Los ambientes cambian constantemente, el *awareness* debe mantenerse actualizado.
- ✓ La gente mantiene su *awareness* interactuando con el ambiente.
- ✓ El *awareness* es generalmente una meta secundaria (en general la meta principal no es mantener el *awareness*, sino completar alguna tarea colaborativa en el ambiente).

Tomemos como ejemplo a una persona caminando a través de la intersección en un pasillo. Si la persona no es consciente de otra persona que está entrando en la intersección, la carencia de tal awareness causará una colisión. Lo mismo ocurrirá en video conferencias o en cualquier trabajo cooperativo mediado por computadora. En el acceso a archivos mediante una red, sin soporte de *awareness* podría causar pérdida de trabajo de un usuario cuando otro intenta editarlo al mismo tiempo. Accidentes de esta naturaleza condujo a las formas más primitivas de *awareness*, por ejemplo, lockear el archivo que está siendo editado. De esta manera a un usuario al que se le denega el acceso a un archivo, toma conciencia que está siendo editado por otro. Por supuesto, que esta forma primitiva tiene muchas restricciones, y limita la comunicación y cooperación.

Basados sobre estas observaciones es fácil ver que el *awareness* es fundamental para la seguridad, el confort y la productividad en diversas actividades sociales y colaborativas, sin tener en cuenta la naturaleza virtual o física del ambiente.

La información disponible en un ambiente físico permite a la gente mantener conocimiento de las ubicaciones de las otras personas, actividades e intenciones relativas a las tareas y al espacio; lo que permite trabajar más eficientemente. Podemos mirar a otra persona para saber qué está haciendo, dónde está trabajando, podemos escuchar el sonido de ciertas herramientas que nos indican qué trabajo se está realizando.

El diseñador de sistemas de groupware debe intentar recrear las condiciones y pistas que permiten a la gente estar conscientes del trabajo del resto de las personas. Desafortunadamente muchas de las cosas que soporta el *awareness* en un escenario cara a cara, como visión periferial, miradas rápidas, sonido tridimensional, posibilidad de ver el ambiente entero, desaparecen en un ambiente de groupware.

El *awareness* se vincula con:

- Control de acoplamiento: Se llama acoplamiento al grado en que las personas trabajan juntas. La gente mientras trabaja en un trabajo en grupo se mueve entre actividades individuales y de grupo, esto es entre acoplamiento débil y fuerte. Dado que ven una oportunidad de colaborar, necesitan discutir o decidir algo con el resto del grupo, necesitan planear la siguiente actividad, o necesitan a otra/s personas del grupo para poder concluir su tarea actual.

- Comunicación: simplifica la comunicación ya que al tener información de las actividades de las personas del grupo, por ejemplo se es consciente de lo que una persona está realizando y se podría conversar sobre el tema en cuestión.

- Coordinación: al igual que en punto anterior, sabiendo que determinada persona del grupo se encuentra modificando un documento, podemos coordinar las acciones para evitar conflictos, es decir hacer que ocurran en un orden correcto y de manera que no afecten el resto de las acciones del grupo.

De todo lo mencionado anteriormente podemos concluir, que el *awareness* ayuda a la gente a moverse entre actividades individuales y compartidas. Provee un contexto en el cual interpretar las expresiones de los otros y referencias deícticas sobre los objetos, permite anticipación de las acciones de los otros y reduce el esfuerzo necesario para coordinar tareas y recursos.

3.3 Mecanismos de recolección y provisión de awareness

Recolección de información de awareness [GG]

Hay tres principales orígenes para recolectar información de *awareness* en una colaboración cara a cara. La gente obtiene información que es producida por los cuerpos de las personas, por los artefactos en el ambiente, y por conversaciones y gestos. Estos mecanismos que la gente usa para recolectar información son llamados comunicación consecuencial, *feedthrough* y comunicación intencional.

- Comunicación intencional

La comunicación intencional es el mecanismo usado a través de gestos y conversaciones. Una conversación entre dos personas puede no incluir a una tercera explícitamente, sin embargo ésta última puede enterarse de lo que están hablando los involucrados en la conversación. Por medio de la comunicación verbal, las personas pueden intercambiar información de *awareness*.

- Comunicación consecuencial

El mecanismo de ver y oír la actividad de las otras personas en el ambiente se llama comunicación consecuencial, la información que surge como consecuencia de la actividad de una persona dentro del ambiente (Segal 1994). Esta clase de comunicación con el cuerpo no es intencional en la manera que lo son los gestos explícitos.

- Feedthrough

Los artefactos en el ambiente son el tercer origen de información de *awareness*. Por sus posiciones, orientaciones y movimientos, los artefactos pueden mostrar el estado de las interacciones de la gente con ellos. Los artefactos también incluyen el ambiente acústico ya que pueden producir diferentes sonidos según la manipulación de las personas. El mecanismo de determinar las interacciones de una persona por ver y escuchar los artefactos se llama feedthrough (Dix et al 1993).

Provisión de información de awareness

Sistemas de CSCW varían en los mecanismos que usan para proveer información de *awareness*. Un mecanismo al cual se refieren como informacional es proveer facilidades explícitas a través de las cuales los miembros del grupo de trabajo o bien de la hipermedia colaborativa informan al resto de sus actividades.

Otro mecanismo al cual se refieren como restrictivo al rol tiene origen en el concepto de roles en sistemas colaborativos; sin embargo este mecanismo provee información sólo acerca del carácter de la actividad y no del contenido.

3.4 El problema del awareness [GG]

En una situación de la vida real donde las personas se encuentran cara a cara, el *awareness* es relativamente fácil de mantener y los mecanismos de colaboración son naturales y espontáneos.

La información disponible en un ambiente físico permite a la gente mantener un conocimiento de las ubicaciones de las otras personas, actividades e intenciones relativas a las tareas y al espacio, lo que permite trabajar más eficientemente. Podemos mirar a otra persona para saber qué está haciendo, dónde está trabajando, podemos escuchar el sonido de ciertas herramientas que nos indican qué trabajo se está realizando, podemos escuchar las diferentes conversaciones que se están realizando, vemos los movimientos, gestos, de cada persona, etc.

Desafortunadamente el *awareness* en un ambiente virtual y en especial en una hipermedia donde los integrantes del grupo de trabajo se encuentran distribuidos, es mucho más difícil de mantener y menos natural.

Existen varias razones que provocan esta dificultad, entre ellas:

- Los dispositivos de entrada y salida usados en el sistema de computación proveen sólo una fracción de la información perceptiva que está disponible en un escenario cara a cara (la información que la gente usa en el mundo real para mantener la pista de los otros). Cada clase de información sensorial está reducida o ausente en un groupware.

- La interacción de los usuarios en un groupware genera mucha menos información que las acciones en un escenario real. Mucha información de *awareness* disponible en el mundo real surge de la manipulación directa de los artefactos, pero la manipulación en aplicaciones de computadora es mucho menos directa que en el mundo real. Notemos la diferencia que existe entre quitar un elemento que se encuentra sobre un escritorio tomándolo con la mano y ubicándolo en otro lugar, y utilizar el mouse para ubicarse sobre el elemento que deseamos eliminar y cliquear sobre el botón correspondiente a la opción de borrar.

El diseñador de sistemas de groupware debe intentar recrear las condiciones y pistas que permitan a la gente mantener un sentido del *awareness*. Lamentablemente, muchas de las cosas que soporta el *awareness* en un escenario cara a cara como visión periferial, miradas rápidas, sonido tridimensional, la posibilidad de ver el ambiente entero, percepciones, etc.; son difíciles de reproducir en un escenario de groupware.

Los diseñadores se enfrentan a los siguientes problemas en el diseño del soporte de *awareness:*

- ¿Qué información debería capturar un sistema de groupware acerca de la interacción de los otros con la hipermedia?
- ¿Cómo debería ser presentada esa información a los otros participantes?
- ¿Cuándo la información de awareness será más útil?

Se deben considerar tanto los elementos que forman el *awareness* de la gente, y el mecanismo que ellos utilizan para recolectar la información de *awareness.*

Esos elementos caen en dos grupos generales: los que tratan sobre qué está pasando con la otra persona, y los que tratan acerca de dónde está pasando.

El soporte de *awareness* en sistemas de groupware puede ser difícil, especialmente cuando los participantes pueden trabajar en diferentes partes de la hipermedia (relaxed -WYSIWIS), en donde cada uno de ellos no pueden directamente ver lo que otros están haciendo, y pueden perder la pista de dónde están en el espacio.

Conclusión

Con todo lo mencionado anteriormente podemos concluir que el *awareness* es útil para hacer las interacciones colaborativas más eficientes, con menos esfuerzo y con menos cantidad de errores. Hay varias actividades colaborativas donde los beneficios del *awareness* son evidentes; por ejemplo podemos mencionar:

- ✓ Reduce el esfuerzo necesario para la comunicación.
- ✓ Permite a las personas anticiparse a las actividades de los otros.
- ✓ Ayuda a coordinar las actividades colaborativas provee un contexto apropiado para ayuda y asistencia.

3.5 Awareness en hipermedias

Para entender por qué es necesario el *awareness* en una hipermedia colaborativa nos referimos a un ejemplo. Si pensamos en un tour por la ciudad de La Plata, recorriendo sus principales atracciones y entretenimientos, podemos observar que si el recorrido por la ciudad lo realizamos con un grupo de personas, la comunicación es fluida, está

nutrida de referencias deícticas y gestuales, es decir se utilizan todas las capacidades físicas y perceptivas que existen en la comunicación entre las personas. Por ejemplo, un turista podría estar participando del tour con un familiar, con el cual podría hacer comentarios, decidir qué lugar visitar luego, podrían separarse por un tiempo para visitar sitios de interés individuales para luego reunirse en un lugar predeterminado, podría interactuar con otros turistas pidiendo consejos sobre qué lugares visitar.

Por otro lado, en un tour virtual representado a través de una hipermedia, se ven reducidas todas las capacidades de comunicación que existen entre las personas en el mundo real, se remueven ciertos movimientos y sonidos, se reduce el campo visual, se pierden las referencias deícticas y gestuales.

En una hipermedia para poder simular el tour real, necesitaríamos ser conscientes de en qué nodos se encuentran los otros usuarios del grupo para poder establecer una comunicación y poder encontrarse con alguien, ser conscientes de quiénes se encuentran realizando el tour para poder tomar decisiones grupales o hacer comentarios.

Otro ejemplo (que posteriormente diseñaremos y analizaremos más profundamente) consiste de una hipermedia colaborativa de la facultad de informática. Consideremos un sitio web de la facultad de informática en donde los usuarios pueden tomar clases virtuales y pueden de manera colaborativa realizar alguna tarea o conversar con otros usuarios.

Contrastándolo con una clase real dentro de la facultad, vemos que en la hipermedia es necesario para que se pueda colaborar eficientemente saber qué otros usuarios se encuentran en el sistema, en dónde están, que están haciendo, entre otras cosas.

Es decir que podemos notar que para los dos ejemplos anteriormente mencionados necesitamos de cierta información de *awareness* para que nos permita colaborar eficientemente.

De esto surge la necesidad de que la hipermedia brinde cierta información referente a la interacción de los usuarios que reemplace la carencia de las características citadas anteriormente. En un tour real o en una clase real, cada persona está viendo cara a cara al resto de los miembros del grupo; en un tour virtual o en una clase virtual esto podría proveerse por ejemplo, con una lista con los nombres de los usuarios, o una foto de cada usuario.

Finalmente podemos definir ***awareness de hipermedia*** como:

"La colección de conocimiento "on line" que una persona mantiene acerca del estado de las interacciones de los participantes con la hipermedia."

Esto incluye el conocimiento acerca de quiénes están en la hipermedia, qué nodos de la hipermedia están navegando, en qué nodos de la hipermedia están trabajando, sobre qué objetos se está trabajando (con esto nos referimos a objetos conceptuales, por ejemplo, en nuestro

tour por la ciudad de La Plata, un objeto conceptual podría ser la Catedral), qué acciones están realizando los usuarios, en qué momento ocurren determinadas acciones, y cuáles son las intenciones futuras de los usuarios.

Recordemos que el *awareness* ayuda a la gente a moverse entre actividades individuales y compartidas, provee un contexto en el cual interpretar las expresiones de los otros y referencias deícticas sobre objetos, permite anticipación de las acciones de los otros, y reduce el esfuerzo necesario para coordinar tareas y recursos.

3.5.1 Awareness y la World Wide Web

Existen dos enfoques diferentes en la comunicación y colaboración basados en web: *comunicación asincrónica* y *comunicación sincrónica*.

La primera situación ocurre cuando diferentes usuarios comparten información a través de una aplicación web sin el requerimiento de tener que estar conectados al mismo tiempo. En ese caso, un usuario no es consciente si hay otros usuarios conectados, se anula el sentimiento de presencia compartida y el grado de interactividad es muy bajo; por lo que la información de *awareness* que se puede brindar es menor.

La situación de comunicación sincrónica ocurre cuando los usuarios se encuentran e interactúan en un ambiente colaborativo multiusuario. En este caso, la información de *awareness* que se puede dar es muy rica y brinda muchos beneficios a las personas que se están comunicando o compartiendo un trabajo.

Para que el valor de la World Wide Web continúe incrementándose para actividades sociales y colaborativas, es necesario diseñar, desarrollar y explorar el uso del *awareness* en las aplicaciones web. Es necesario realizar un análisis para entender cuándo y cómo proveer *awareness* es apropiado para una aplicación, actividad o situación determinada.

La cantidad de información de *awareness* que uno hace disponible a los usuarios se incrementa con el valor de la actividad social y colaborativa. Por ejemplo, si una persona está trabajando en un importante proyecto dentro de un grupo, entonces hacer al resto del grupo consciente de la disponibilidad, o del estado de su trabajo en el proyecto, es cómodo y muy valioso. Sin embargo, permitir a 30 millones de usuarios de la web, que no forman parte de un grupo de trabajo o que no tienen un objetivo común, que sean conscientes del estado de disponibilidad o del trabajo de uno, probablemente no tenga tantos beneficios y a veces puede llegar a incomodar el trabajo.

3.6 Elementos de Awareness

Basándonos en los elementos de workspace awareness propuestos en [GG] pensamos qué información puede interesarle a cada usuario de una hipermedia acerca del resto de los participantes. Por ello proponemos

diferentes elementos de *awareness*, los cuales proveen un vocabulario básico para pensar acerca de los requerimientos de *awareness*.

La siguiente tabla muestra un conjunto de elementos de *awareness* que consideramos relevantes en una hipermedia colaborativa; y una lista de preguntas asociadas que un usuario podría hacerse durante una actividad compartida.

Muchos de los elementos se pueden categorizar dentro de dos grandes grupos; aquellos que tratan con *qué* está pasando con otras personas, y aquellos que tratan con *dónde* está pasando.

Elemento de Awareness	**Pregunta relevante**
Presence	¿quiénes están en la hipermedia?
LocalPresence	¿quiénes están en un nodo determinado de la hipermedia?
UserLocation	¿en qué nodo se encuentra un usuario determinado?
Density	densidad de usuarios en cada nodo de la hipermedia
UserAction	¿qué acción está haciendo el usuario en la hipermedia?
NodeAction	¿qué acción se está haciendo en un nodo de la hipermedia?
UserHistory	¿cuál es la historia de navegación de un usuario?
UserNavigationIntent	¿dónde navegará luego un usuario?
UserPastAction	¿qué acciones realizó un usuario?
UserBookmarks	¿cuáles son los bookmarks de un usuario?
UserExpectations	¿qué acción se considera que hará luego un usuario considerando el estado actual?
ObjectsUsers	¿qué objetos están siendo usados y por quién?
ObjectHistory	¿qué cambios se han realizado sobre el objeto y quién lo realizó?

Capítulo 4

Diseño de Awareness Orientado a Objetos en Hipermedia Colaborativa

En el capítulo 2, sección 2.4.2 vimos el modelo colaborativo propuesto en la metodología de diseño CHDM. Nuestro objetivo es extender dicho modelo para contemplar más profundamente el awareness, concepto explicado en el capítulo anterior, que muestra su importancia en la colaboración.

Como resultado obtenemos un modelo de diseño de awareness orientado a objetos para especificar el awareness en una hipermedia colaborativa, el framework de awareness como una posible implementación del modelo, y diagramas de interacción para detallar las características dinámicas.

4.1 Modelo de diseño de Awareness

Una vez realizados los pasos de la metodología de diseño OOHDM y los correspondientes a la metodología CHDM es necesario, para que la hipermedia colaborativa contemple el awareness, se realice el siguiente modelo.

Tengamos en cuenta que en este momento el diseñador ya tendrá el modelo conceptual, navegacional y parte del colaborativo. Sólo le falta agregarle las características de awareness.

En esta sección se muestran las clases participantes y sus relaciones, pero no cómo interactúan entre ellas.

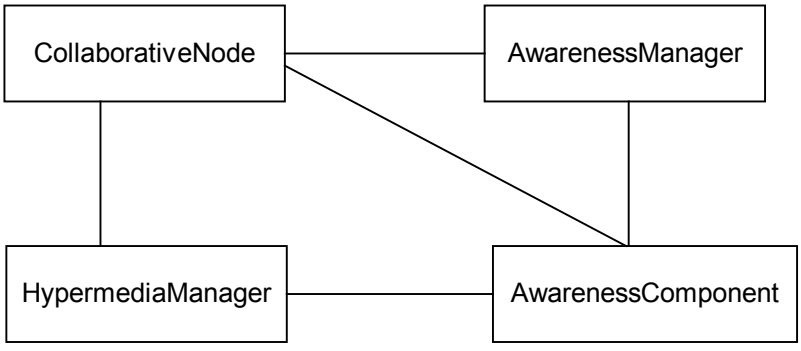


Figura 4.1

Cada nodo colaborativo que desee brindar información de awareness debe crear un objeto de tipo AwarenessComponent indicando qué tipo de awareness quiere contemplar (más adelante en este capítulo veremos cómo indicar el tipo de awareness).

Existe un único AwarenessManager encargado de recibir las notificaciones de eventos que suceden en los nodos colaborativos y reenviarlos a todos los AwarenessComponent de los nodos colaborativos respectivamente.

Cada AwarenessComponent (implementa el patrón de diseño Composite, [GHJV94-pag163]) se encargará de solicitar al HypermediaManager mediante su respectivo nodo colaborativo la información necesaria si el evento ocurrido involucra al/los tipo/s de awareness que éste contemple.

El HypermediaManager es un subsistema que conoce a toda la hipermedia.

Cabe destacar que este modelo de awareness debe unirse al modelo colaborativo presentado en el capítulo 2 , sección 2.4.2. Es decir la clases CollaborativeNode y AwarenessManager son las mismas en ambos modelos.

4.2 Framework de Awareness

Proponemos como una posible implementación del awareness en una hipermedia colaborativa un framework que llamamos ACH (Awareness in Collaborative Hypermedia). Como veremos más adelante dicho framework implementa determinadas clases del modelo recientemente presentado.

En la siguiente sección explicamos brevemente qué es un framework, para luego presentar el framework ACH propuesto.

4.2.1 Definición de Framework

El tema del reuso de trabajo ya realizado por otros no es nuevo en el área del desarrollo de software. [ACM01]

Una tecnología de reuso ideal provee componentes que pueden ser fácilmente conectados para realizar un nuevo sistema. El desarrollador del software no necesita conocer cómo están implementados. Ejemplos de esto pueden ser los patterns y los components, sin embargo en la actualidad toma fuerza el concepto de frameworks.

Desde el punto de vista de estructura podemos definir al framework como: "*un diseño reusable de todo un sistema o de una parte de un sistema que está representado por un conjunto de clases abstractas y por la forma en que sus instancias interactúan* "; si lo definimos describiendo su propósito podemos decir: " *es el esqueleto de una aplicación que puede ser personalizado por un desarrollador de la aplicación* ". Los frameworks son una técnica de reuso orientada a objetos.

Los frameworks son los generadores de aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.

Los frameworks deben generar las aplicaciones para un dominio entero. Por lo tanto, debe haber puntos de flexibilidad que se puedan modificar para ajustarse a una aplicación particular.

Al diseñar un framework es importante identificar los puntos flexibles también llamados *hot-spots*. Los hot-spots son las clases o los métodos abstractos que deben ser implementados por el usuario del framework.

Las tres etapas principales del desarrollo del framework son análisis del dominio, diseño del framework, y la instanciación del framework.

Durante el análisis del dominio se definen los requisitos del dominio y se contemplan posibles extensiones futuras. En la fase de diseño se definen las abstracciones, se modelan los hot-spots y frozen-spots (puntos del framework que quedarán "congelados", son los pedacitos del código puestos en ejecución ya dentro del framework que llaman a uno o más hot-spots proporcionados por el ejecutor). Por último en la fase de instanciación se implementan los hot-spots.

4.2.2 Framework de Awareness - ACH

A partir del modelo de awareness previamente explicado consideramos la necesidad de diseñar un framework de awareness que implemente determinadas clases de dicho modelo.

Proponemos el siguiente framework de awareness como una posible implementación del awareness para una hipermedia colaborativa.

El análisis de requerimientos realizado para diseñar el modelo de awareness entonces, cubre la etapa de análisis de dominio del desarrollo del framework.

Diseño

La clase AwarenessManager explicada en la sección 4.1 forma parte del framework de awareness.

Además el framework propuesto está formado por dos jerarquías diferentes. Una jerarquía representa los diferentes tipos de awareness que, en nuestro caso, los nodos colaborativos pueden tener; es decir detalla a la clase AwarenessComponent explicada en la sección anterior. Una segunda jerarquía representa los tipos de eventos de awareness que pueden ocurrir, esta jerarquía se corresponde directamente con la mencionada anteriormente y será explicada a continuación.

Es importante destacar que el desarrollador de una aplicación que quiera usar este framework debe utilizar ciertas interfases a partir de las cuales ocurre la interrelación entre todos los componentes de la aplicación y el framework de awareness. Dichas interfases también serán mencionadas en esta sección.

Jerarquía de Componentes de Awareness

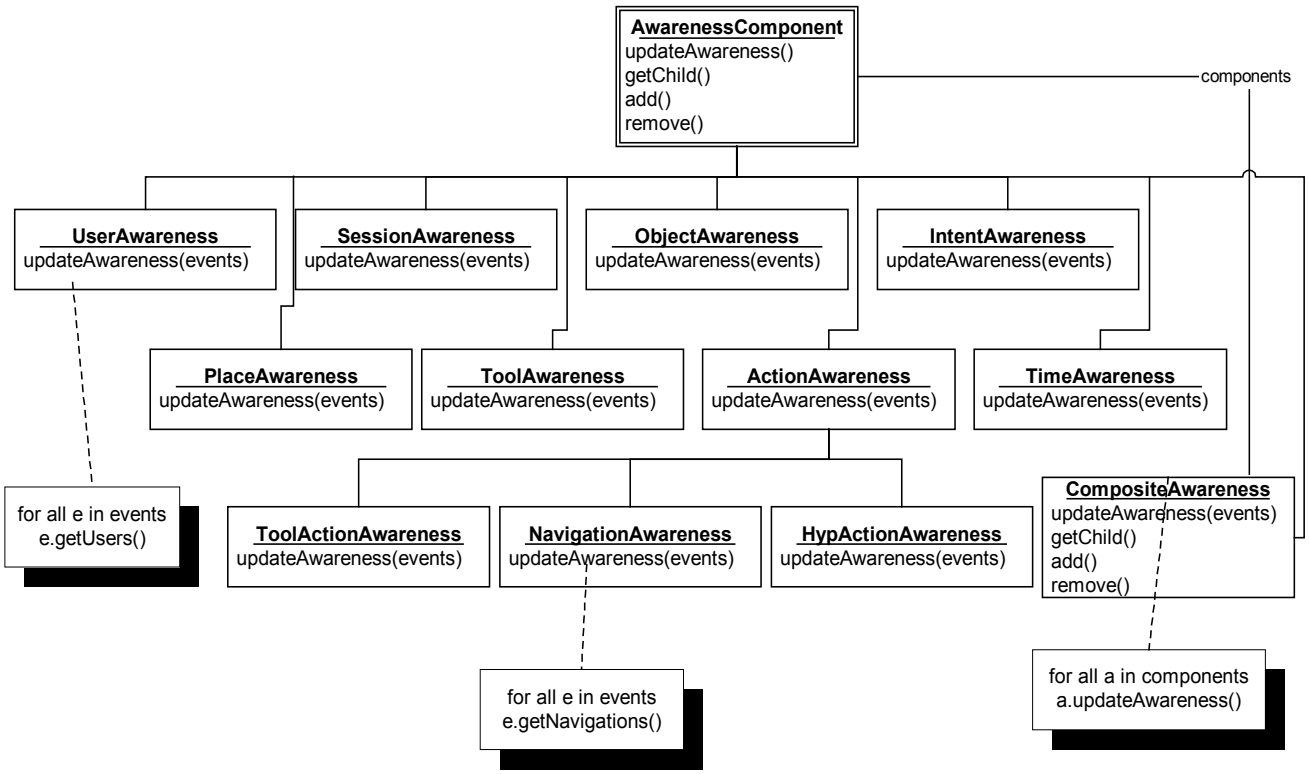


Figura 4.2

Esta jerarquía representa los diferentes tipos de awareness que un nodo colaborativo puede crear. Es decir representa los diferentes tipos de información referente a la interacción de los usuarios con la hipermedia.

Para esta jerarquía se utilizó el patrón de diseño orientado a objetos Composite [GHJV94-pag163].

Por ejemplo, si un nodo colaborativo quiere contemplar awareness de usuario y de lugar, para saber quiénes están y en qué nodo de la hipermedia, debe crear un componente de awareness compuesto, es decir un CompositeAwareness, formado por los componentes individuales UserAwareness y PlaceAwareness.

Nótese que, si el día de mañana necesitamos contemplar en nuestras aplicaciones colaborativas un nuevo tipo individual de componente de awareness, esta jerarquía permite hacerlo simplemente agregando una clase nueva dentro de la jerarquía, sin tener que realizar ningún otro cambio.

Al utilizar el patrón Composite tanto los componentes compuestos como los individuales tienen un protocolo común (*updateAwareness()* en

nuestro caso), lo cual permite que siempre se interactúe con los AwarenessComponent de manera uniforme.

Jerarquía de eventos de Awareness

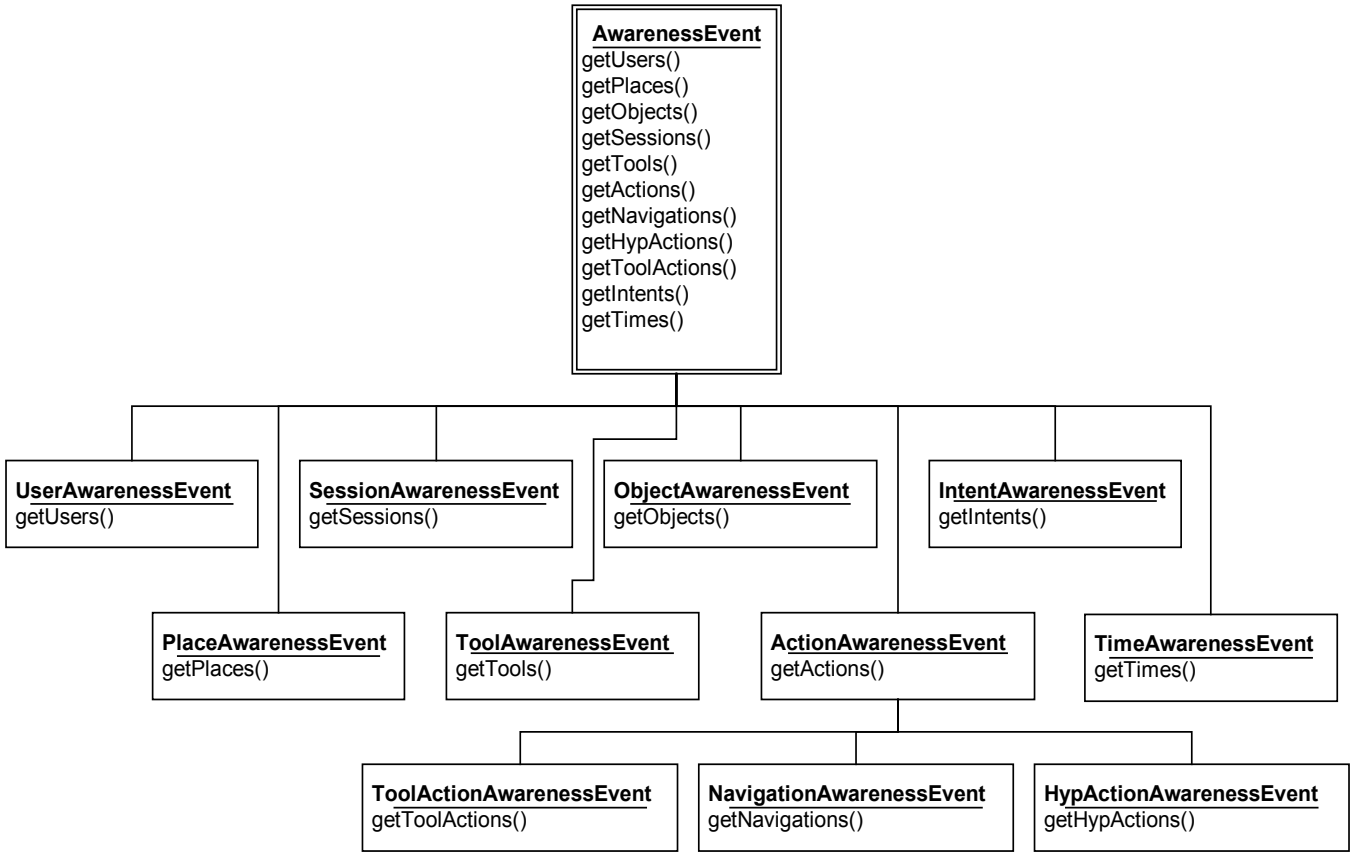


Figura 4.3

Esta jerarquía representa los diferentes tipos de eventos de awareness y tiene una correspondencia directa con la jerarquía de componentes de awareness vista anteriormente.

La clase AwarenessEvent es la raíz de la jerarquía y contiene todos los métodos referentes a todos los tipos de eventos. Estos métodos realizan el comportamiento default (en principio no harían nada).

Cada clase en el segundo y tercer nivel de la jerarquía sobrescribe el método correspondiente a su tipo; por ejemplo la clase UserAwarenessEvent sobrescribe el método *getUsers*(), si a una instancia de esta clase se le envía por ejemplo el mensaje *getPlaces*(), al no sobrescribirlo , se ejecutará el método de la clase AwarenessEvent que tiene el comportamiento default. Si por el contrario se le envía el mensaje *getUsers*() se ejecutará el método sobrescrito por dicha clase.

La utilidad de estos eventos sirve para que los AwarenessComponent sepan si deben ignorar o no un evento ocurrido.

Interfases

A partir del framework de awareness explicado anteriormente, es necesario que los nodos colaborativos y el HypermediaManager respondan a ciertos mensajes, para que todo funcione correctamente.

Para esto proponemos las siguientes interfases que deberían implementar aquellas clases del modelo que correspondan a los nodos colaborativos y al HypermediaManager.

Interfase IAwareNode >>

getUsers()
getObject()
getSessions()
getTools()
getActions()
getHMActions()
getNavigations()
getIntents()
requestChat()

Interfase IAwareHypManager >>

getUsers()
getPlaces()
getObjects()
getSessions()
getTools()
getNavigations()
getHMActions()
getActions()
getIntents()
navigate()
requestChat()
sendMail()
joinSession()

4.2.3 Diagramas de interacción

En esta sección mostraremos con diagramas de interacción cómo interactúan las clases del framework y las del modelo.

Los diferentes diagramas muestran distintos escenarios de colaboración que pueden ocurrir en la hipermedia colaborativa.

En la figura siguiente mostramos el diagrama de interacción que grafica la sucesión de mensajes enviados entre los objetos del framework de awareness generados a partir de la ocurrencia de un evento x en la hipermedia colaborativa.

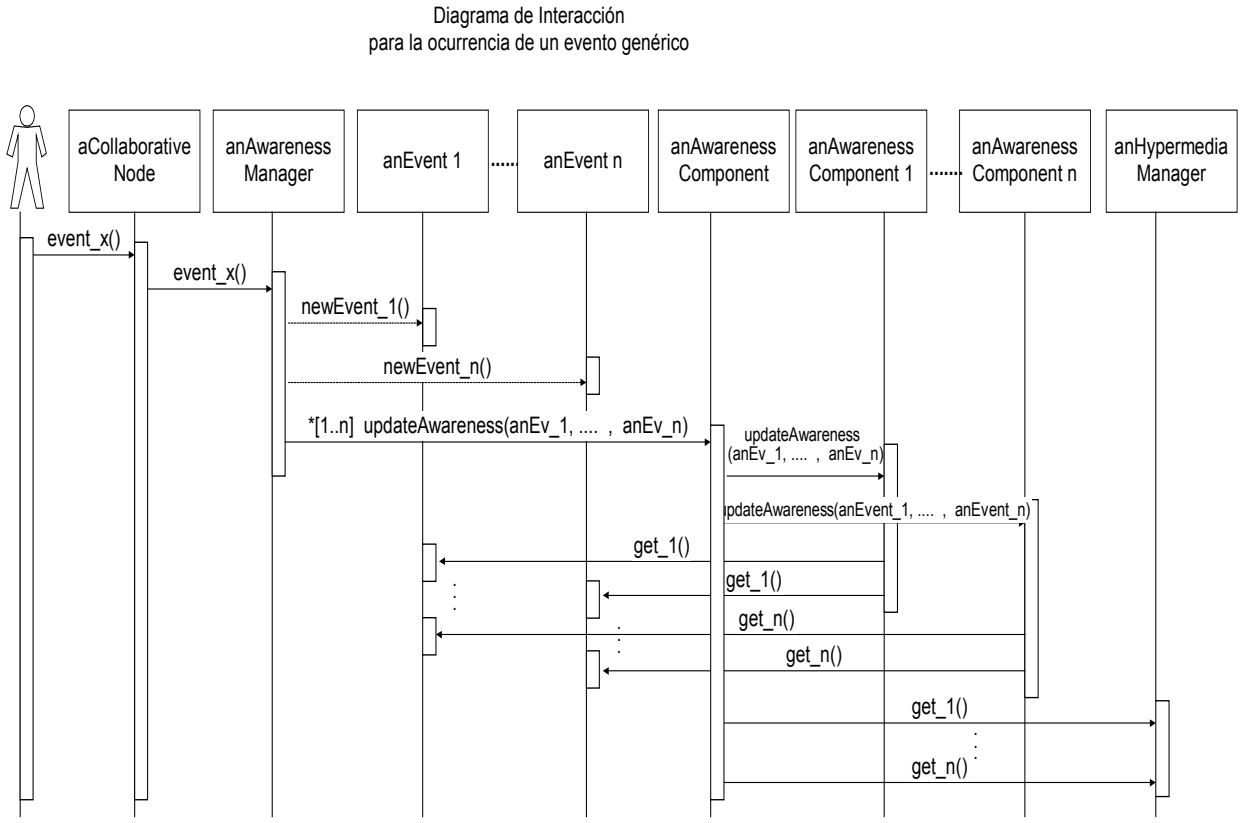


Figura 4.4

Suponemos un nodo colaborativo genérico, es decir, sin indicar el tipo de componente de awareness, y la ocurrencia de un evento cualquiera en la hipermedia colaborativa.

El AwarenessManager una vez que es notificado de que un evento x ha ocurrido en la hipermedia colaborativa, es el encargado de crear las instancias de eventos de awareness que se vean involucrados en el evento generado en la hipermedia.

Cabe destacar que un evento ocurrido en la hipermedia puede implicar la creación de n eventos de awareness; por ejemplo, el ingreso de un usuario en un nodo colaborativo (lo cual genera un único evento en la hipermedia) implica los eventos de awareness de usuario y de lugar y posiblemente de tiempo, dado que el hecho de que un usuario entre en un nodo afectará a la información de awareness tanto de usuario (quién entra en el nodo), como de lugar (en qué nodo entra el usuario), también si se está implementado awareness de tiempo afectará a la información de tiempo del nodo.

Luego, el AwarenessManager le envía a cada uno de los AwarenessComponents la notificación de actualización de awareness, mediante el método updateAwareness, pasándole como parámetro los n eventos de awareness creados previamente.

A partir de este punto cada AwarenessComponent actuará de diferente manera dependiendo de sus componentes de awareness. Recordemos que la jerarquía de componentes de awareness tiene la estructura del pattern Composite. De acuerdo al comportamiento del pattern, se le envía un método *updateAwareness()* (con los eventos como parámetro), a cada uno de sus componentes hojas. Cada componente hoja del Composite debe interactuar con cada uno de los eventos recibidos solicitándoles, sin tener en cuenta el tipo de evento de awareness, la información de su interés.

El AwarenessComponent entonces, sabe qué información solicitarle a la HipermediaManager. Una vez que recibe la información solicitada se visualizan las actualizaciones.

Para comprender mejor la interacción existente entre estas dos jerarquías y el resto del modelo veamos el siguiente ejemplo.

Consideraciones:

- Supongamos que ingresa un usuario en la hipermedia, entonces todos los nodos que contemplen awareness de usuario y de lugar deberían refrescar su información.

- Consideramos que el ingreso del usuario ya fue registrado por la aplicación y sólo mostraremos la interacción ocurrida luego para actualizar la información de awareness.

Una vez registrado el ingreso del usuario, el nodo colaborativo en donde ocurrió el evento es el encargado de notificarle al AwarenessManager de lo ocurrido. El AwarenessManager se encarga de crear el/los evento/s de awareness correspondientes, en este caso creará un evento UserAwarenessEvent y un PlaceAwarenessEvent, luego notificará a todos los AwarenessComponent de los nodos el/los evento/s ocurridos a través del mensaje *updateAwareness(anEvents)* (información de los eventos creados). Si bien todos los AwarenessComponent serán notificados de lo ocurrido, a no todos les interesará esa información. Serán los AwarenessEvent los responsables de indicar a los AwarenessComponent si deben o no recolectar la información de awareness.

Una vez que cada AwarenessComponent es notificado que el evento ocurrido en la hipermedia, en nuestro ejemplo “entrada de un usuario”, es de su interés, se comunicará con el HipermediaManager a través de su correspondiente nodo para solicitarle la información.

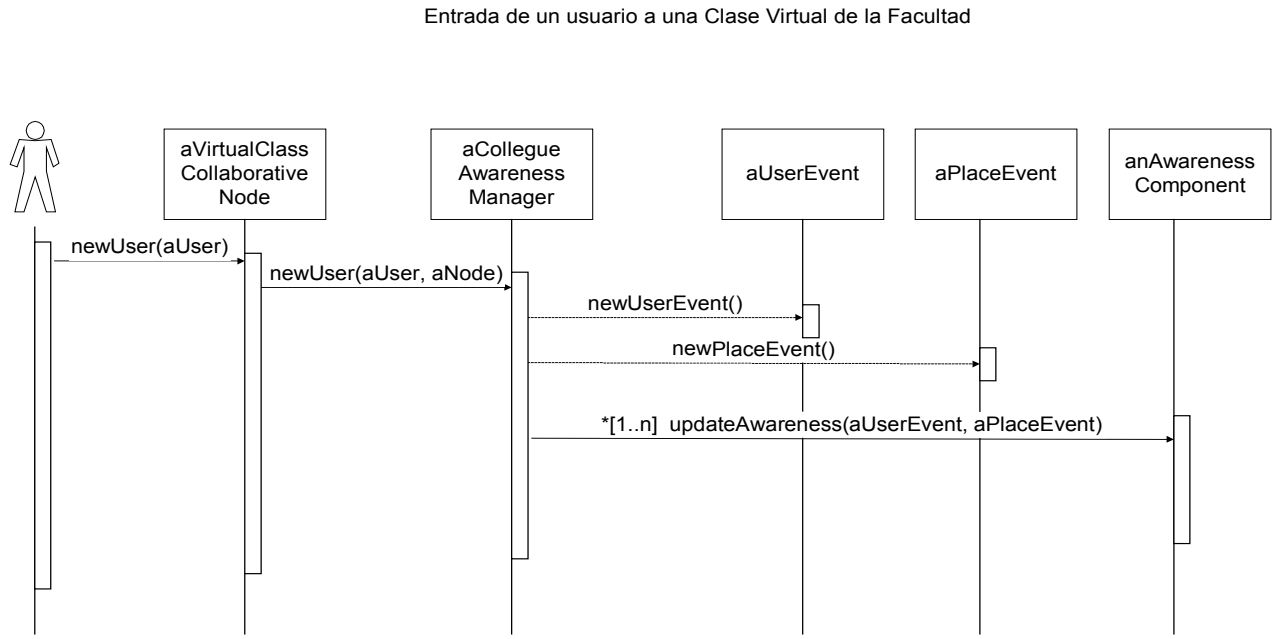


Figura 4.5

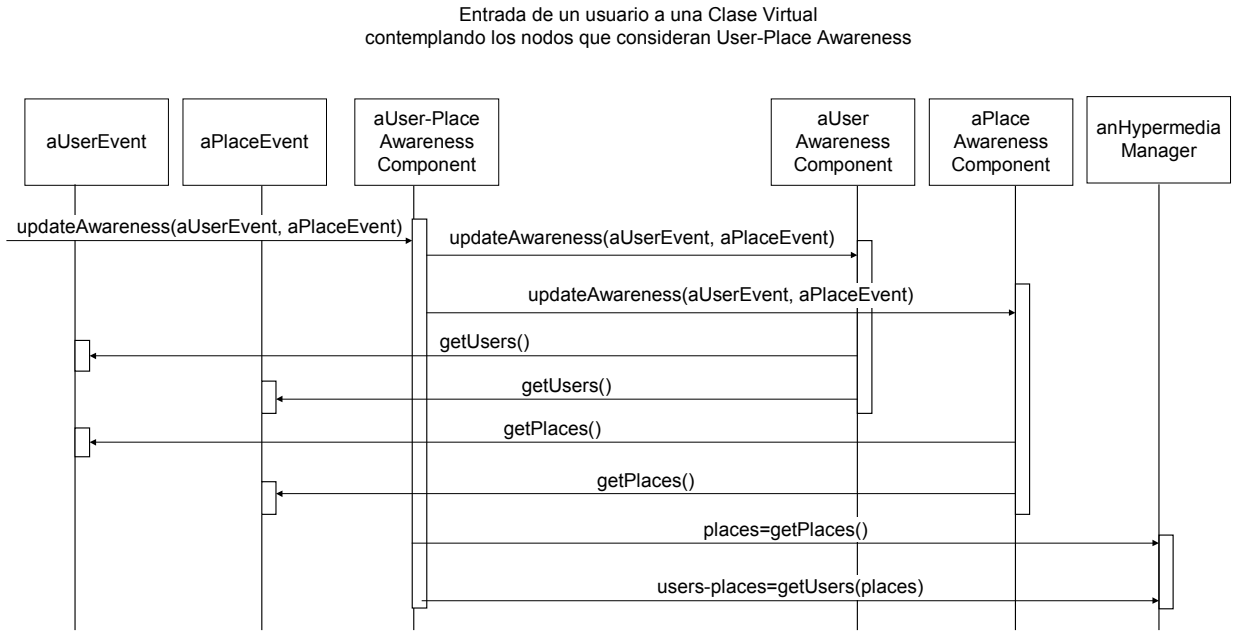


Figura 4.6

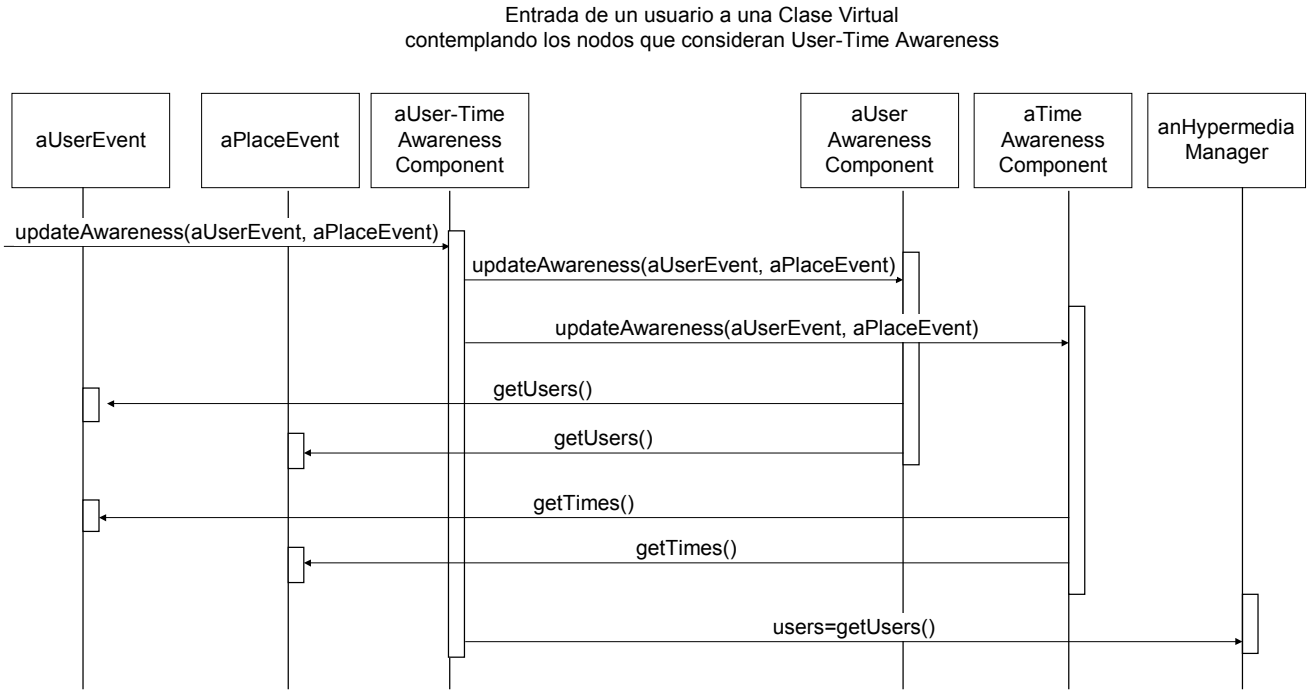


Figura 4.7

4.2.4 Relación entre eventos que ocurren en la Hipermedia y eventos generados por el AwarenessManager

La siguiente tabla muestra los eventos de awareness generados por el AwarenessManager para cada evento ocurrido en los nodos colaborativos de la hipermedia.

Cada evento ocurrido en la hipermedia brinda nueva información de awareness. Entonces es necesario que el AwarenessManager genere los eventos de awareness, definidos en la jerarquía de eventos de awareness, correspondientes a la información brindada de manera que los cambios se vean reflejados en todos los nodos colaborativos.

Es importante destacar que un evento cualquiera ocurrido en la hipermedia afecta a información concerniente a diferentes, incluso pueden ser más de uno, tipos de awareness.

Es necesario que el AwarenessManager cree instancias de las clases de eventos de awareness involucradas en el evento ocurrido en la hipermedia, para que todos los nodos colaborativos vean la información de awareness actualizada correctamente.

Por ejemplo, en la tabla vemos que el evento de hipermedia ocurrido cuando un usuario navega (*navigateToNode*) implica que el AwarenessManager genere los eventos de awareness UserEventAwareness, PlaceEventAwareness y NavigationEventAwareness; debido a que la navegación de un usuario desde un nodo origen (NO) a un nodo destino (ND) afecta a la ubicación del usuario ya que deja de estar en el nodo NO para aparecer en el nodo

ND, como así también a la historia de navegación de dicho usuario que debe incluir esta última navegación como parte de la historia.

Como puede notarse en la tabla, el evento de awareness TimeAwarenessEvent siempre es generado por el AwarenessManager, proponemos que esto sea así de manera de tener registrado el momento en el que ocurre cualquier acción en la hipermedia.

Evento de Hipermedia en el nodo colaborativo	Eventos de Awareness generados por el AwarenessManager
NewUser	UserAwarenessEvent, PlaceAwarenessEvent, TimeAwarenessEvent
LeftUser	UserAwarenessEvent, PlaceAwarenessEvent, TimeAwarenessEvent
navigateToNode	UserAwarenessEvent, PlaceAwarenessEvent, NavigateAwarenessEvent,TimeAwarenessEvent
joinSession	UserAwarenessEvent, SessionAwarenessEvent, IntentAwarenessEvent, TimeAwarenessEvent
leftSession	UserAwarenessEvent, SessionAwarenessEvent, IntentAwarenessEvent, TimeAwarenessEvent
objectChanged	UserAwarenessEvent, ObjectAwarenessEvent, TimeAwarenessEvent
joinTool (*)	UserAwarenessEvent, ToolAwarenessEvent, TimeAwarenessEvent
LeftTool	UserAwarenessEvent, ToolAwarenessEvent, TimeAwarenessEvent
toolActionPerformed	UserAwarenessEvent, TimeAwarenessEvent ToolActionAwarenessEvent
addBookmark	UserAwarenessEvent, HypAwarenessEvent TimeAwarenessEvent
removeBookmark	UserAwarenessEvent, HypAwarenessEvent TimeAwarenessEvent
downloadFile	UserAwarenessEvent, HypAwarenessEvent TimeAwarenessEvent
DoFTP	UserAwarenessEvent, HypAwarenessEvent TimeAwarenessEvent
FillForm	UserAwarenessEvent, HypAwarenessEvent TimeAwarenessEvent
requestJoinTool	Cuando ocurre este evento en un nodo colaborativo, este se lo pasa al hipermediaManager quien genera 2 joinTool (*)
SendEmail	

4.3 Características del modelo

El modelo de awareness propuesto ofrece la posibilidad de que cada nodo colaborativo tenga más de un componente de awareness. Esta necesidad surge debido a que pueden existir casos en que la información de awareness requerida no tenga relación entre sí, o que el desarrollador desee modelarla separadamente; entonces no tendría sentido que esa información se muestre en un mismo AwarenessComponent.

Para entender mejor lo dicho recientemente supongamos que en un determinado nodo colaborativo deseamos información de usuarios con las herramientas y objetos que cada uno está utilizando. No tiene sentido relacionar las herramientas y los objetos porque representan conceptos independientes; lo que es interesante poder modelar es usuarios y objetos, y por otro lado, usuarios y herramientas; por lo cual sería conveniente que el nodo colaborativo tenga dos AwarenessComponent, uno que represente la información usuario-herramienta (es decir qué herramienta está utilizando cada usuario) y otro que represente la información usuario-objeto (qué objeto está manipulando cada usuario).

En la siguiente figura se muestra la porción correspondiente al awareness del modelo colaborativo del ejemplo mencionado anteriormente.

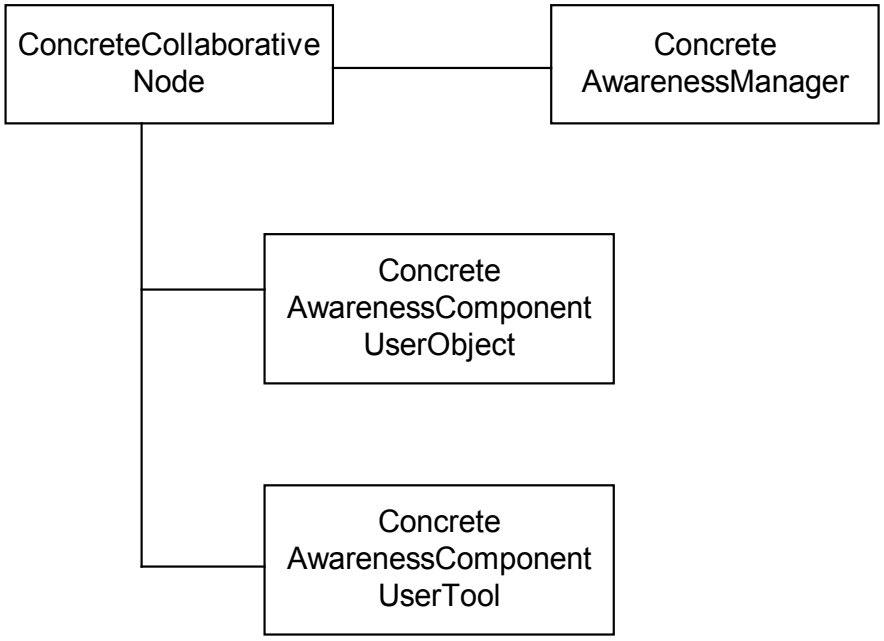


Figura 4.8

El diseño de awareness propuesto brinda la posibilidad de combinar cualquier tipo de awareness. Si bien esto otorga la libertad de diseñar componentes de awareness de manera de que se adapten a cualquier tipo de aplicación de hipermedia colaborativa con diferentes requerimientos; queda a criterio del desarrollador diseñar las combinaciones de awareness de manera que la información brindada sea relevante, teniendo en cuenta los usuarios y el contexto de la aplicación.

4.4 Características del framework

Destacamos lo simple que es extender el framework propuesto para agregar un nuevo tipo de awareness.

Dos puntos deben tenerse en cuenta:

- ✓ Se deberá agregar una subclase de AwarenessComponent (en la primer jerarquía mencionada anteriormente), en donde por formar parte del patrón Composite dicha clase deberá implementar el método *updateAwareness()*.
- ✓ Se deberá agregar una sublcase de AwarenessEvent (en la segunda jerarquía) relativa al nuevo componente agregado, esta clase deberá implementar el método de su interés dependiendo de para qué sirve el nuevo tipo de awareness que se está agregando.

Otro punto importante a destacar en el framework es que si se desea cambiar el tipo de awareness que contempla un nodo colaborativo concreto, solamente será necesario cambiar la instancia del AwarenessComponent en el nodo.

Al estudiar la importancia del awareness en trabajos grupales vimos interesante y beneficioso poder proveer funcionalidad al sistema a partir de la información de awareness brindada por los nodos colaborativos. Esto permite que la colaboración y comunicación se realice de manera más flexible, directa y natural.

Pensemos en la posibilidad de que a partir de un componente de awareness que muestra a los usuarios logueados, podamos invitar a algunos de ellos a un chat, simplemente cliqueando sobre la imagen o icono que represente al usuario. Pensemos en la posibilidad de unirse a una sesión de una clase virtual colaborativa que nos resultó de interés al ver mediante la información de awareness que está siendo mediada por un determinado profesor, pudiendo hacerlo directamente desde la información de awareness.

Con el framework de awareness propuesto brindamos la posibilidad de que se lleve a cabo el tipo de funcionalidad mencionada anteriormente de la misma manera que si se llevara a cabo mediante la aplicación y no desde el awareness de la misma.

Proponemos las siguientes funciones que se pueden iniciar a partir de la información de awareness:

- RequestChat: invitar a otro usuario a chatear a partir de la información de awareness de usuario.
- SendEmail: mandar un e-mail a un usuario a partir de la información de awareness de usuario.
- Navigate: navegar a otro nodo a partir de la información de awareness de lugar.
- JoinSession: unirse a una sesión a partir de la información de awareness de sesión.
- JoinTool: unirse a una herramienta colaborativa a partir de la información de awareness de herramienta.

Estas funciones forman parte del protocolo de la interfaz definida para los nodos colaborativos.

Esquema de prioridades de composición de Awareness

1. Place
2. User
3. Tool
4. Object
5. Session
6. Navigation
7. HypAction
8. ToolAction
9. Intent
10. Time

Con el framework de awareness propuesto se tiene la libertad de combinar de varias formas distintas los tipos de awareness. Esto genera que el componente de awareness compuesto le solicite al HypermediaManager esa información combinada, lo que sobrecargaría al protocolo del HypermediaManager con métodos que contemplen todas las combinaciones posibles.

Por esta razón proponemos métodos simples en el HypermediaManager (*getUsers(), getPlaces()*, etc; en lugar de *getUsersOfPlace(), getPlaceOfUser()*, etc) que reciben como parámetro una lista genérica de manera que:

```
/* retorna todos los usuarios de la hipermedia */
- getUsers( [] )


/* retorna los usuarios que se encuentran en cada nodo de la hipermedia*/
- getUsers( Places[] )
      for all places p in Places
          p.getUsers()
```

En el framework además usamos un esquema de prioridades (listado anteriormente), que indica el orden en el que se debe solicitar la información desde los AwarenessComponent al HypermediaManager. Decidimos usar el mencionado esquema para evitar que el HypermediaManager tenga que recolectar información que luego será filtrada por posteriores solicitudes dado que el requerimiento de información es combinado. Para ilustrar esto mejor veamos un ejemplo:

Si hiciéramos *getUsers(Actions[])* siguiendo el criterio anterior nos devolvería primero todas las acciones que se realizaron en la hipermedia, y luego por cada una de esas acciones quién la realizó, sin tener en cuenta si dichas acciones pertenecen a usuarios que siguen estando en la hipermedia o ya se han ido. Esto no debería ser así, dado que si algún nodo está interesado en información de awareness de acción-usuario, es natural pensar que la información que desea brindar el nodo colaborativo

es qué acción está realizando cada usuario online; para lograr obtener dicha información, utilizando el esquema de prioridades mencionado llamaríamos al método *getActions(Users[]),* este método retorna por cada usuario online en la hipermedia que acciones realizó.

4.5 Relaciones entre los modelos y el framework

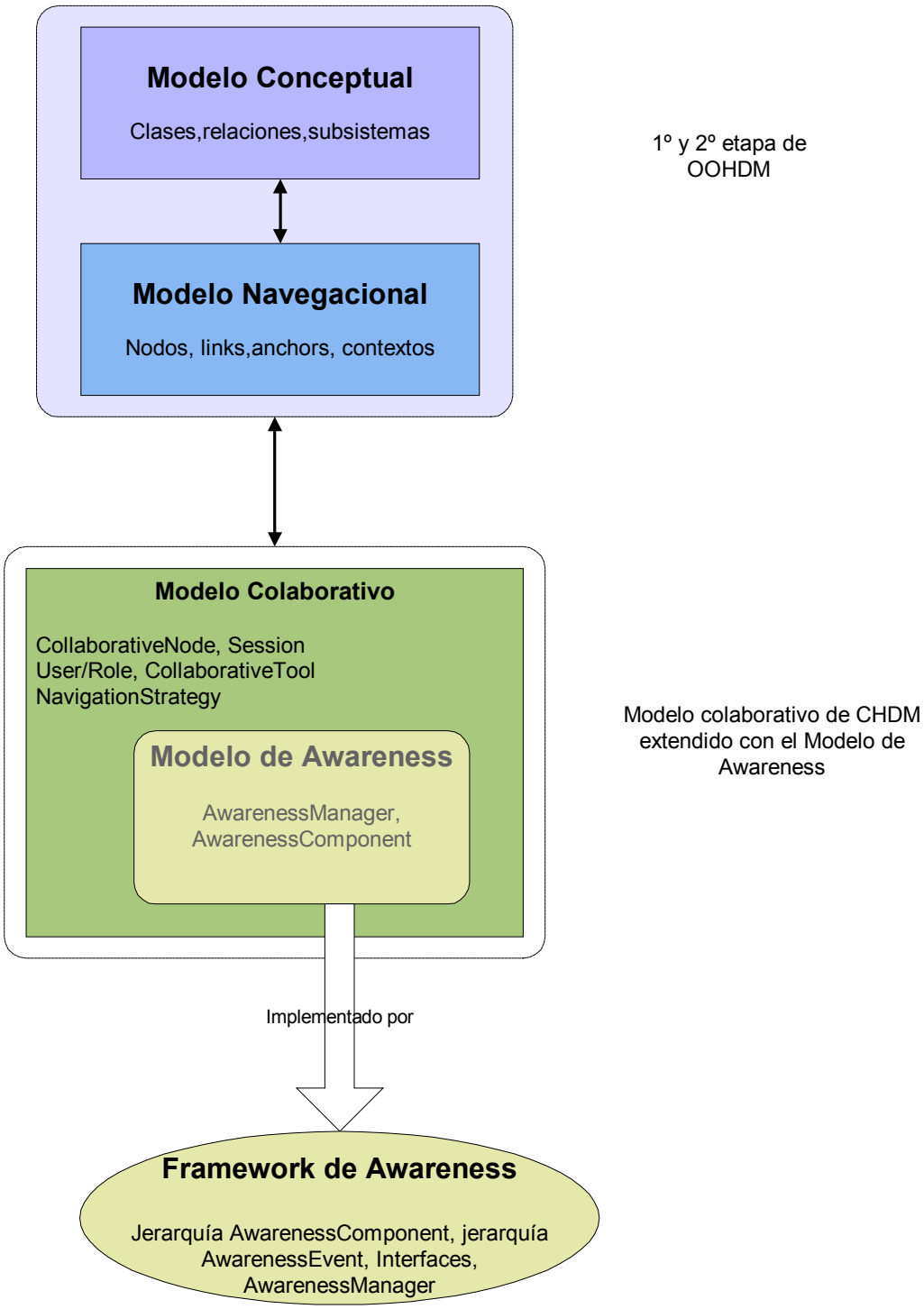


Figura 4.9

Utilizando la metodología de diseño OOHDM comenzamos con la etapa de modelo conceptual en donde el resultado de esta etapa es un esquema de clases y objetos construido a partir de subsistemas, clases y relaciones. Luego pasamos a la segunda etapa de OOHDM, es decir al diseño navegacional en donde surgen las nociones de nodos, links, anchors; se construyen las visiones navegacionales del modelo conceptual.

Más tarde pasamos al modelo colaborativo de CHDM , éste tiene en cuenta ciertas características de las aplicaciones colaborativas tales como identificación y manejo de los usuarios de la aplicación y sus roles, sesiones de colaboración entre ellos junto con la manera en que interactúan y colaboran, control de concurrencia, awareness, uso de herramientas colaborativas sincrónicas y asincrónicas, etc. La etapa del diseño colaborativo se especifica posteriormente al diseño del modelo navegacional y consiste en identificar los nodos de la aplicación a los que se les agregará alguna de dichas características colaborativas y modelar nodos colaborativos para ellos.

Si bien CHDM permite diseñar aplicaciones de hipermedia colaborativa, no fue el objetivo de dicha metodología hacer un análisis profundo del Awareness.

En este trabajo de grado presentamos un Modelo de Awareness que puede ser utilizado para extender CHDM. En el Modelo de Awareness aparecen el AwarenessManager y AwarenessComponent como clases fundamentales. Además diseñamos e implementamos un framework de awareness que aplicamos para la construcción del prototipo, en este framework aparecen las dos jerarquías anteriormente explicadas junto con las interfases propuestas.

Capítulo 5

Prototipo *InCoWeb* Análisis, Diseño e Implementación

En este capítulo comenzamos con el desarrollo, etapas de análisis, diseño e implementación, de un ejemplo que consta de una hipermedia colaborativa de la facultad de informática que brinda información de awareness.

Aplicamos todo lo definido y explicado en los capítulos previos. Instanciamos el modelo conceptual de OOHDM (cap. 2, sección 2.4.1), el modelo colaborativo de CHDM (cap. 2, sección 2.4.2) y el de awareness presentado en esta tesis (cap. 4), como también mencionamos cómo fue utilizado el framework de awareness ACH (cap. 4).

A partir del modelo de awareness y el framework de awareness propuestos en este trabajo de grado, diseñamos e implementamos un prototipo utilizando ambos.

La intención de realizar el prototipo es mostrar concretamente cómo ayuda la información de awareness en aplicaciones que tenga aspectos de colaboración en la web.

Prototipo *InCoWeb* – Análisis y Diseño

El prototipo consiste de una hipermedia colaborativa de la facultad de informática teniendo en cuenta la visión del alumno. Al prototipo lo llamamos InCoWeb, Informática Colaborativa en la Web. Esta incluye nodos que forman clases virtuales de la carrera como también nodos que definen otras áreas importantes de la facultad como la biblioteca, ventanilla de alumnos, buffet, cátedra, etc.

Para elegir la aplicación a implementar, analizamos dentro de las posibles aplicaciones colaborativas aquellas en las que claramente pudiéramos mostrar la necesidad y los beneficios de contar con información de awareness. Elegimos desarrollar una aplicación de la facultad de Informática centrándonos sólo en los aspectos específicamente colaborativos; dejando de lado los aspectos informativos de la facultad, donde la información de awareness no sería necesaria, ni brindaría beneficios, incluso, podría llegar a molestar, distraer, confundir al usuario ya que navegar en un sitio con contenidos informativos resulta

ser una actividad principalmente individual en donde tener conocimiento acerca de otros usuarios no es importante, ni necesario.

Rescatamos los aspectos colaborativos que puede tener una aplicación de este tipo. Consideramos una aplicación en donde los usuarios puedan colaborar, en este caso en donde los alumnos puedan tomar clases de las diferentes materias mediante la aplicación. En este punto notamos los beneficios de contar con información de awareness ya que vemos útil que los alumnos que están tomando una misma clase tengan conocimiento de quiénes más se encuentran presentes, de la presencia online del profesor y la posibilidad entonces de comunicarse, como también saber qué se encuentra haciendo determinada persona.

5.1 Funcionalidad del prototipo

El objetivo del prototipo es mostrar la utilidad del modelo de awareness en el diseño, y del framework ACH (cap. 4 , sec. 4.2.2) en la etapa de implementación. En este punto cabe mencionar que nuestra intención fue destacar las características colaborativas, y en particular el awareness; sin pretender brindar la funcionalidad completa que pudiera tener una aplicación web colaborativa de una facultad, ni tampoco hacer hincapié en los aspectos de la interfase visual.

En InCoWeb cada alumno deberá loguearse en la aplicación para poder ingresar al sistema. Luego de esto dependiendo de las clases a las que esté habilitado cada alumno el sistema brindará la posibilidad de navegar a dichas clases, a la biblioteca o al buffet. Las materias en la aplicación están representadas por varias clases virtuales a las que sólo pueden acceder los alumnos inscriptos. Cada usuario puede elegir el orden en el cual acceder a las distintas clases.

Nuestro prototipo provee la visión del alumno de la facultad colaborativa. Consideramos que todos los alumnos ya han sido dados de alta en la aplicación con las materias en las que cada uno está inscripto.

Dejamos de lado la visión de administración la cual tendría todas las funcionalidades para realizar el mantenimiento de alumno, materias, profesores, etc.

Teniendo en cuenta que el prototipo es para mostrar el uso del modelo y framework de awareness propuesto la aplicación mostrará información de awareness.

En particular mostraremos: qué nodos forman parte del sistema, qué usuarios se encuentran logueados en el sistema, cuál es la historia de navegación de cada usuario, en cuál nodo se encuentra cada usuario, trabajando con qué herramienta está cada usuario (si estuviera en alguna), en qué momento (fecha y hora) navegó cada usuario a diferentes nodos.

Otra característica importante a destacar es que la información de awareness mostrada también brindará la posibilidad de a partir de dicha información participar en alguna herramienta colaborativa que se encuentre presente en el sistema (por ejemplo invitar a chatear), como también navegar a determinado nodo.

5.2 Diseño del prototipo

Para el diseño del prototipo utilizamos los modelos conceptual y navegacional de OOHDM (cap. 2, sección 2.4.1), el modelo colaborativo de CHDM (cap. 2, sección 2.4.2) con la extensión del modelo de awareness presentado en esta tesis (cap. 4).

A continuación presentamos los diferentes modelos:

5.2.1 Modelo Conceptual

El diseño conceptual es un modelo del dominio de la aplicación. Como mencionamos anteriormente, nuestro objetivo no fue realizar una aplicación completa de la facultad, por lo tanto en el modelo conceptual sólo mostramos las principales clases intervinientes.

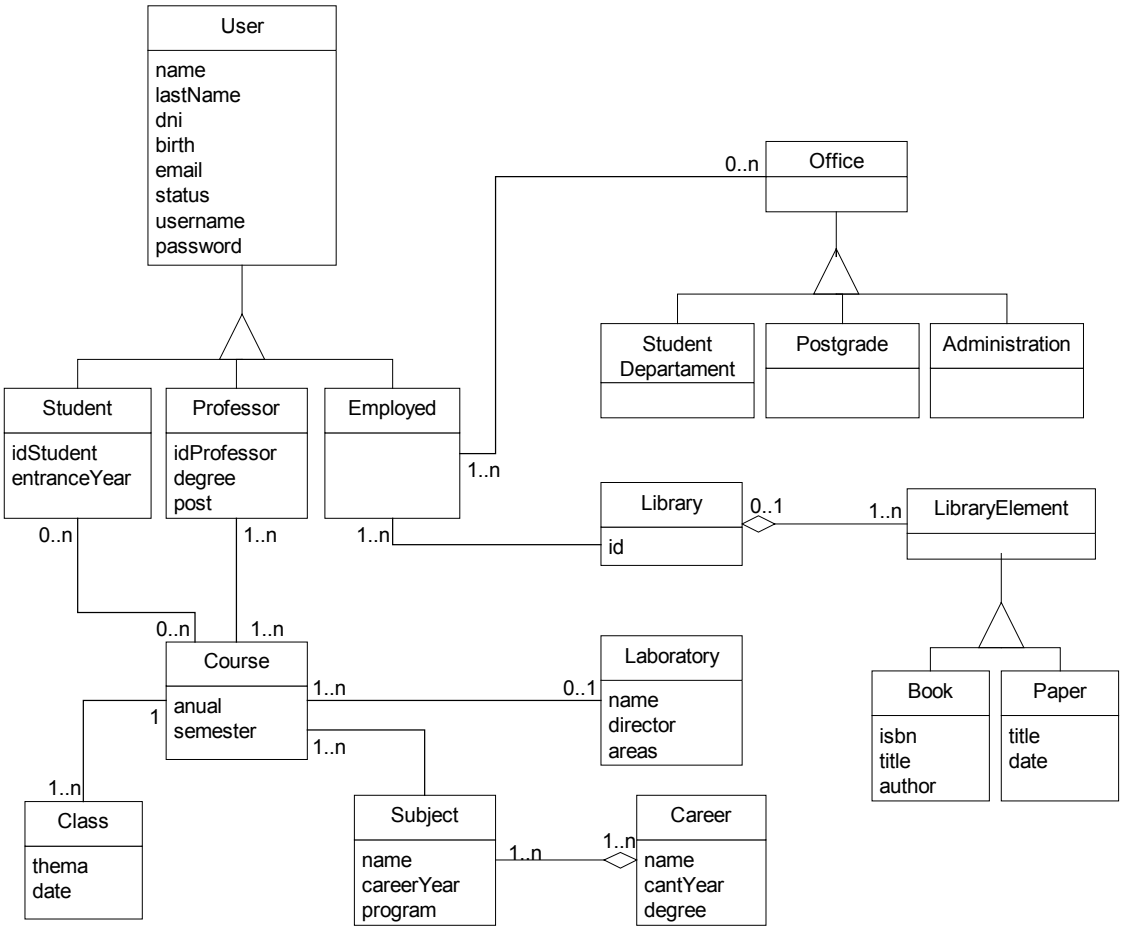


Figura 5.1 - Modelo Conceptual

Definimos las clases Employed, Professor y Student como los usuarios del dominio, también vemos la clase Library para representar la biblioteca que pueda ser consultada por los usuarios, la clase Course para

representar las cátedras de la facultad, dicha clase está relacionada con la clase Subject para representar las materias y con la clase Class para representar las clases de las diferentes materias. En la facultad de Informática existen laboratorios de Investigación, representados por la clase Laboratory, como también diferentes oficinas administrativas, representadas por las distintas subclases de la clase Office.

5.2.2 Modelo Navegacional

En OOHDM una aplicación es vista como una visión navegacional del modelo conceptual. Si bien el diseño de la navegación se expresa en dos esquemas, el esquema de Clases Navegacionales y el esquema de Contextos Navegacionales, aquí sólo presentaremos el esquema de contextos navegacionales para tener una visión preliminar de la navegación del prototipo.

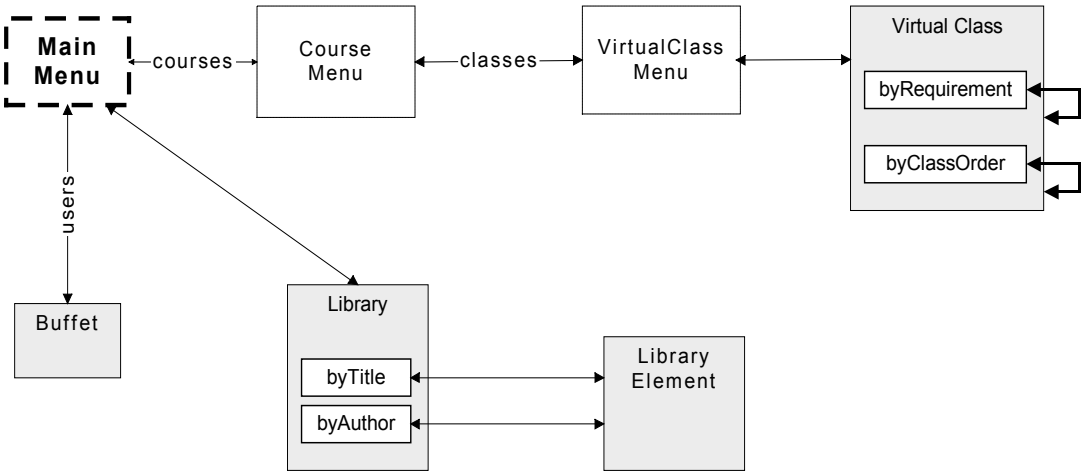


Figura 5.2 – Modelo Navegacional

En el modelo navegacional del prototipo InCoWeb, aparece un nodo navegacional al cual llamamos *Main Menu* que es un índice desde el cual podemos navegar al nodo *Buffet*, al nodo *Library*, y al nodo que contiene un índice con las materias en las que está inscripto un determinado alumno o profesor, que lo llamamos *Course Menu.*

A partir del índice *Course Menu* podemos acceder a diferentes índices, *Virtual Class Menu*, los cuales contienen por cada materia, un listado con las diferentes clases virtuales de la materia. Es decir, a partir del índice *Virtual Class Menu* podemos navegar a los nodos navegacionales de las clases virtuales, llamado *Virtual Class*. El nodo *Virtual Class* contiene dos contextos navegacionales; *byRequirement* (clases virtuales por requeriemiento) y *byClassOrder* (clases virtuales por orden secuencial).

El nodo navegacional *Library* contiene contextos navegacionales, *byTitle* y *byAuthor,* mediante los cuales accedemos a *Library Element*

(nodo que representa libros, papers, etc.) por título y por autor respectivamente.

5.2.3 Modelo Colaborativo

En el modelo colaborativo se modelan los aspectos colaborativos de la aplicación. En este modelo se identifican los nodos de la aplicación a los que se les agregará alguna característica colaborativa y se modelan nodos colaborativos para ellos. El awareness se encuentra modelado utilizando el modelo de awareness presentado en el Capítulo 4, integrándolo al modelo colaborativo de CHDM.

Los nodos colaborativos considerados en InCoWeb tienen diferentes niveles de colaboración y diferentes tipos de awareness, dependiendo de la funcionalidad de cada nodo en particular.

Toda la información de awareness que se muestre en la aplicación dependerá de con qué componentes de awareness fue configurado cada nodo colaborativo de la hipermedia.

Consideramos que la información de awareness que un usuario quisiera ver en el nodo buffet de la aplicación no es la misma que quisiera ver en un nodo de alguna materia, por ejemplo.

En el prototipo consideramos los siguientes nodos con awareness:

Para el nodo Home definimos awareness de usuario y de history. Esto permite, al tener awareness de usuario tener conocimiento de qué usuarios forman parte del sistema, sabiendo además si se encuentran online o no; y al tener awareness de history cualquier usuario puede saber cuáles fueron las últimas navegaciones de cada usuario, esta información permite entre otras cosas delinear las diferentes estrategias de cursar las materias de cada alumno, como también sacar el estadísticas de navegación o hacer análisis sobre el tipo de las navegaciones.

Para los nodos colaborativos que representan las materias definimos awareness de usuario, de lugar y de herramienta. Es importante desde el punto de vista de un usuario que está cursando una materia, ser consciente de los usuarios que están en la misma materia para intercambiar información, si el profesor está conectado para realizarle consultas mediante un chat propio de esa materia, etc.

Para el nodo Buffet definimos awareness de lugar. Nos pareció importante saber quiénes se encuentran dentro del buffet

En la figura siguiente se muestra una vista del modelo colaborativo correspondiente a una clase virtual de una materia de la Facultad de Informática. Cabe destacar que el modelo colaborativo presentado sólo corresponde a una clase virtual del prototipo, y no es el modelo colaborativo completo, ya que cada nodo colaborativo puede tener características de colaboración diferentes dependiendo del escenario modelado. Es decir existe un modelo colaborativo para cada nodo colaborativo.

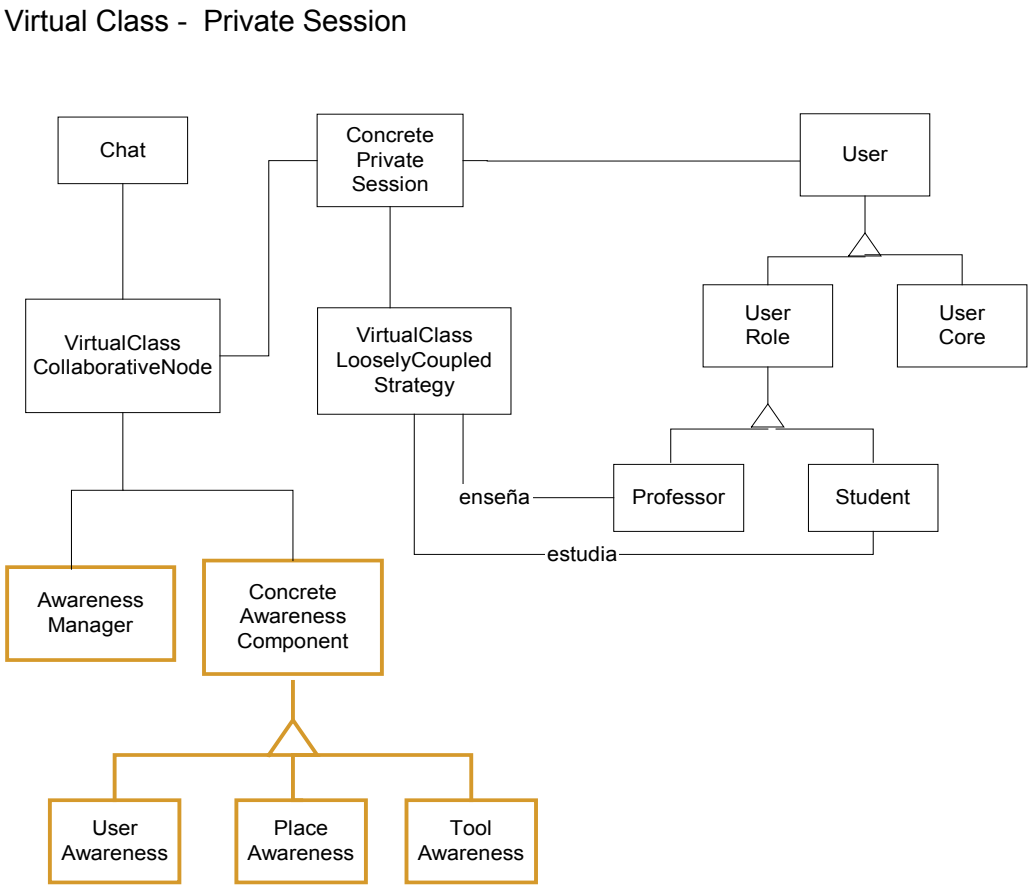


Figura 5.3 - Modelo Colaborativo

Se definió el nodo colaborativo *VirtualClassCollaborativeNode* que representa a una clase virtual a partir de la cual se puede iniciar una sesión colaborativa. Se modela una clase virtual durante una sesión con una estrategia de navegación no acoplada, *VirtualClassLooselyCoupledStrategy,* es decir, en donde los usuarios de la aplicación pueden navegar independientemente del resto de los usuarios. En la sesión encontramos usuarios con rol de *Professor* que puede encontrarse presente en las clases virtuales para que los alumnos los consulten y usuarios con rol de *Student* que asisten a dichas clases. La clase virtual tiene herramientas colaborativas, en este ejemplo, disponemos de un *Chat*, al cual pueden acceder los alumnos y profesores que se encuentren en la clase virtual.

Las clases mencionadas anteriormente son instancias del modelo colaborativo de CHDM. A continuación mencionamos las instancias de las clases del modelo de awareness.

El nodo tiene asociado un *AwarenessManager* encargado de administrar y enviar la información de awareness que le interese al nodo. En nuestro caso, el nodo colaborativo de la clase virtual tiene un *ConcreteAwarenessComponent* que está compuesto por *UserAwareness,*

PlaceAwareness y *ToolAwareness*. Esto implica que el *AwarenessManager* mantendrá informado al nodo acerca de los eventos de awareness relacionados a usuarios, nodos y herramientas colaborativas de la aplicación.

5.3 Relacionando los modelos conceptual, navegacional y colaborativo con awareness

En la figura 5.4 mostramos la relación que existe entre los modelos conceptual y navegacional de OOHDM, y del modelo colaborativo de CHDM con la extensión de awareness presentada en esta tesis.

En el modelo conceptual resaltamos las principales clases del prototipo propuesto, la facultad de Informática. En el modelo navegacional mostramos cómo las clases del modelo conceptual se convierten en nodos navegacionales. Tenemos en cuenta la visión del alumno. En InCoWeb hay un nodo inicial, (que no se corresponde con ninguna clase del modelo conceptual) desde el cual podemos navegar hacia las diferentes materias a las que un determinado usuario esté inscripto. Como así también podemos navegar hacia el nodo buffet y al nodo biblioteca. Desde el nodo de una materia podemos navegar hacia los nodos de las clases virtuales.

Para plantear el modelo colaborativo, primero deben identificarse los nodos a los que se les agregará alguna funcionalidad colaborativa. Esos nodos se presentarán como nodos colaborativos. En nuestro ejemplo, podemos ver que el nodo *VirtualClass* se convierte en un nodo colaborativo, debido a que es un nodo al que acceden grupos de usuarios que pueden colaborar y comunicarse entre ellos y con los profesores. Además provee información de awareness, lo que permite a los usuarios ser conscientes de lo que ocurre en su ambiente de trabajo.

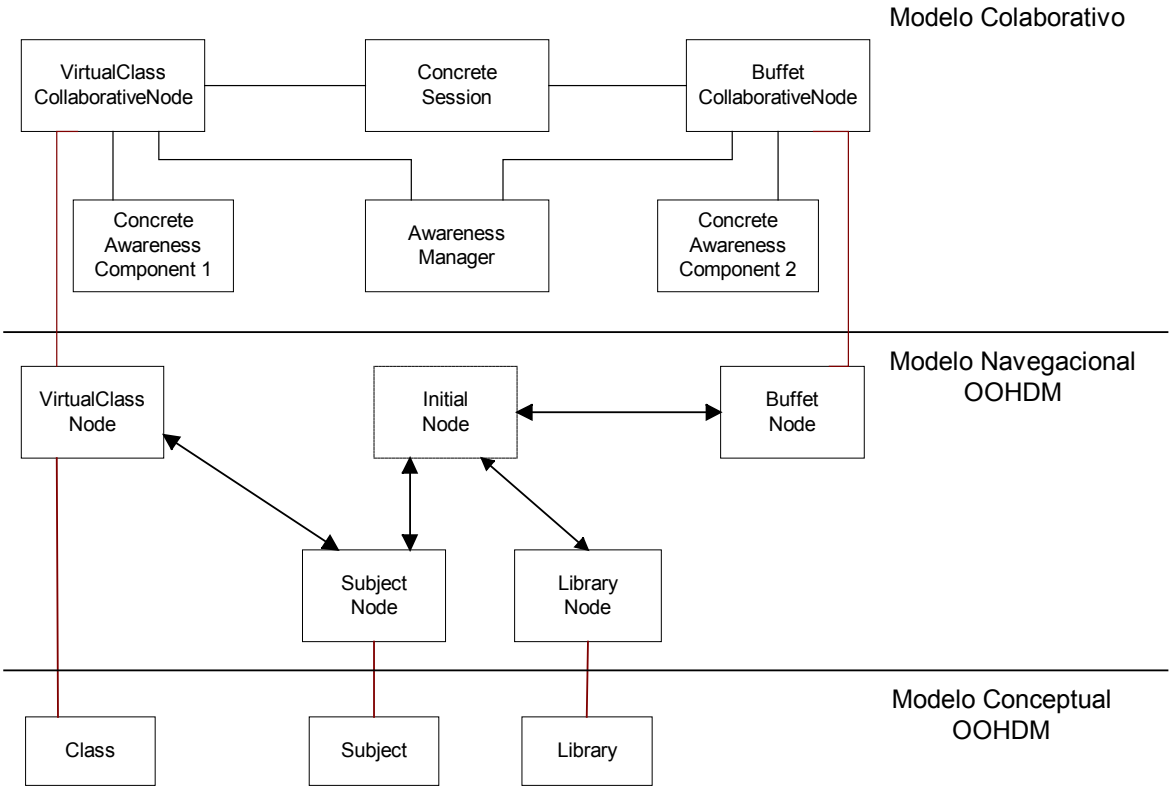


Figura 5.4

El prototipo InCoWeb brinda a partir de la información de awareness la posibilidad de navegar a clases virtuales en las que se encuentre algún usuario. La información de awareness mostrada es la representación visual del componente de awareness del nodo en cuestión y por lo tanto aparece recién en el modelo colaborativo. Por otro lado, debido a la funcionalidad de nuestro prototipo, la cual permite navegar a partir del awareness, se podría pensar en el componente de awareness como un nodo navegacional. Pero debido a que la metodología OOHDM no considera aspectos colaborativos, es un punto a estudiar cómo y dónde debería modelarse la navegación a partir de un componente de awareness.

5.4 Instanciando el framework de awareness ACH en el prototipo

Para la implementación del awareness en el prototipo, utilizamos el framework de awareness ACH (cap. 4, sec 4.2.2).

A partir de las clases del diseño, aquellos nodos colaborativos para los que se necesita información de awareness, es necesario que:

- ✓ Implementen la interfase IAwareNode del framework ACH, esta interfase es la mencionada en el capítulo 4 , sección 4.2.2 y recordemos que su función es permitir la relación de la clase colaborativa que la implementa y las clases del ACH.

- ✓ Contener una instancia de la clase AwarenessManager de ACH. Esta instancia es usada cada vez que ocurre algún evento en la hipermedia para delegar el manejo necesario para almacenar/recuperar la información de awareness.

- ✓ Contener una instancia de la clase AwareHypManager para la obtención de la información de awareness, esta clase implementa una interfase definida en ACH, IAwareHypManager (capítulo 4, sección 4.2.2), que define los métodos necesarios para obtener la información de awareness. La Hipermedia entonces se comunica con esta clase solicitándole la información.

La implementación de los métodos definidos en la interfase IAwareNode es responsabilidad del desarrollador, pero debe tener en cuenta que dichos métodos deben utilizar la instancia del AwarenessManager para enviarle a éste el mensaje correspondiente al evento ocurrido, de manera que el AwarenessManager inicie su proceso de recolección y almacenamiento de la información de awareness que ha sido modificada por el evento (esto último es responsabilidad entera del framework ACH).

Las clases AwarenessManager y AwareHypManager implementan el pattern Singleton [GHJV94-pag 127], de manera que existe una instancia única de dichas clases compartida entre el resto de las clases.

Es importante mencionar que el framework ACH nos sirve para desligar al implementador de una hipermedia colaborativa con características de awareness de todo lo referente a la recolección y almacenamiento de la información de awareness.

El framework es el responsable de generar los eventos de awareness correspondientes al evento ocurrido en la hipermedia y almacenar la información de awareness obtenida para que luego la aplicación colaborativa la utilice.

El framework también contiene las clases necesarias para guardar la información de awareness.

5.5 Extendiendo nuestra aplicación

En esta sección nuestro objetivo es mostrar lo necesario para extender o modificar nuestra aplicación por modificaciones en los tipos de awareness que los nodos colaborativos consideran.

Si lo que necesitamos es agregar o eliminar tipos de awareness en los nodos colaborativos, pero que dichos tipos se encuentran presentes en el framework, lo que es necesario hacer es:

Desde el punto de vista del diseño, sólo es necesario incluir en el modelo colaborativo los nuevos AwarenessComponents ha desarrollar.

Desde el punto de vista de la implementación:

- ✓ Crear uno o más, dependiendo de la necesidad, AwarenessComponent de los tipos de awareness necesarios.

- ✓ Incorporar el o los nuevos AwarenessComponents como instancias del nodo colaborativo correspondiente. Sólo haciendo esto, el framework ya estará notificado de la modificación y comenzará a procesar la información de awareness para el nodo en cuestión.

Por otro lado teniendo en cuenta que la aplicación ha sido desarrollada como en nuestro prototipo, habrá que incluir (si no existieran ya) en la clase AwareHypManager los métodos necesarios para obtener la información de awareness almacenada por el framework.

Podemos notar entonces que una modificación como la planteada, no tiene ningún impacto en el framework, y muy poca modificación en la aplicación.

5.6 Extendiendo el Framework ACH

En esta sección queremos mostrar las modificaciones necesarias por requerimientos de nuevos tipos de awareness no tenidos en cuenta a la hora de implementar el framework.

Desde el punto de vista del diseño del framework, es necesario:

- ✓ Ampliar la jerarquía de componentes de awareness incluyendo en la estructura del patrón Composite los nuevos awareness. Obviamente estos nuevos AwarenessComponents deben definir el método *updateAwareness()*

- ✓ Ampliar la jerarquía de eventos de awareness, incluyendo para cada tipo de awareness nuevo agregado en la jerarquía anterior, un tipo de evento de awareness. Los nuevos eventos incluidos deberán definir los métodos de su interés; dichos mensajes deberán ser incluidos también en la superclase de eventos (la clase AwarenessEvent).

Desde el punto de vista de la implementación, es necesario:

- ✓ Implementar cada una de las nuevas clases que aparecen en la jerarquía de componentes de awareness, implementando por supuesto el método *updateAwareness()*. La implementación de este método no tiene ninguna consideración especial, basta con hacer una copia del *updateAwareness()* de alguno de los ya definidos AwarenessComponents y hacer unos pocos cambios.

Las nuevas clases implementadas deben ser subclases de la clase AwarenessComponent.

- ✓ Implementar cada uno de los nuevos eventos de awareness que aparecen en la jerarquía de eventos de awareness, implementando para cada clase el método de su interés. Incluir cada uno de estos métodos en la superclase AwarenessEvent.

- ✓ Incluir en la clase EventsTable los eventos de awareness que deberán ser creados cuando un evento en la hipermedia genere información del nuevo tipo de awareness que se está incluyendo en el framework.

- ✓ Incluir en la clase AwarenessRegistry un atributo para almacenar la información de awareness del nuevo tipo.

- ✓ Si el nuevo tipo de awareness, también define un nuevo tipo de evento en la hipermedia, entonces debe incluirse en la clase AwarenessManager un método relativo a dicho evento.

Capítulo 6

Prototipo *InCoWeb* –Tecnologías

En este capítulo explicamos brevemente la arquitectura del prototipo InCoWeb.

Mencionamos cuáles fueron las tecnologías que utilizamos para desarrollarlo.

Comenzamos explicando lo que es una aplicación web, ya que nuestro prototipo lo es, cómo está formada, y cuál es la forma correcta de desarrollarla. Explicamos y comparamos dos diferentes modelos usados para desarrollar una aplicación web.

Luego explicamos brevemente las tecnologías usadas, JSP/Servlets, Struts, JDO. Todas estas tecnologías son orientadas a objetos, lo cual permite mantenernos dentro del mismo paradigma durante el análisis y diseño y posteriormente en la implementación.

6.1 Aplicación Web

Una aplicación Web es una colección de componentes individuales que una vez juntos forman una aplicación completa que puede ser instalada y ejecutada por un contenedor web. Los componentes son ligados juntos debido al hecho de que residen en el mismo contexto web y en muchos casos pueden referirse al otro directa o indirectamente.

Una aplicación Web puede ser instalada y ejecutada en múltiples contenedores web concurrentemente. También múltiples instancias de la misma aplicación web pueden ser instaladas en el mismo contenedor web. Sin embargo debido a la manera en la cual las URLs son matcheadas a la correcta aplicación web, cada aplicación web debe tener un nombre único dentro del contenedor web.

Elementos de una aplicación Web

Si bien cada aplicación web tiene sus propios requerimientos y usa determinados recursos, en general todas consisten de uno o más de los siguientes elementos:

- Servlets
- Páginas JSP
- JavaBeans y clases utility

- Documentos HTML
- Archivos multimedia
- Applets, archivos de estilos y javascript
- Documentos de texto
- Meta información que une los elementos anteriores

Arquitectura Web

Muchas aplicaciones y especialmente aplicaciones J2EE, pueden ser descriptas en términos de niveles. La funcionalidad de la aplicación está separada a través de estos niveles funcionales para proveer separación de responsabilidad, reuso, escalabilidad. Esta separación puede ser física, donde cada nivel está ubicado sobre distintos recursos de hard; o puede ser lógica, en este caso uno o más niveles son ubicados en el mismo recurso físico y la separación existe en términos de software.

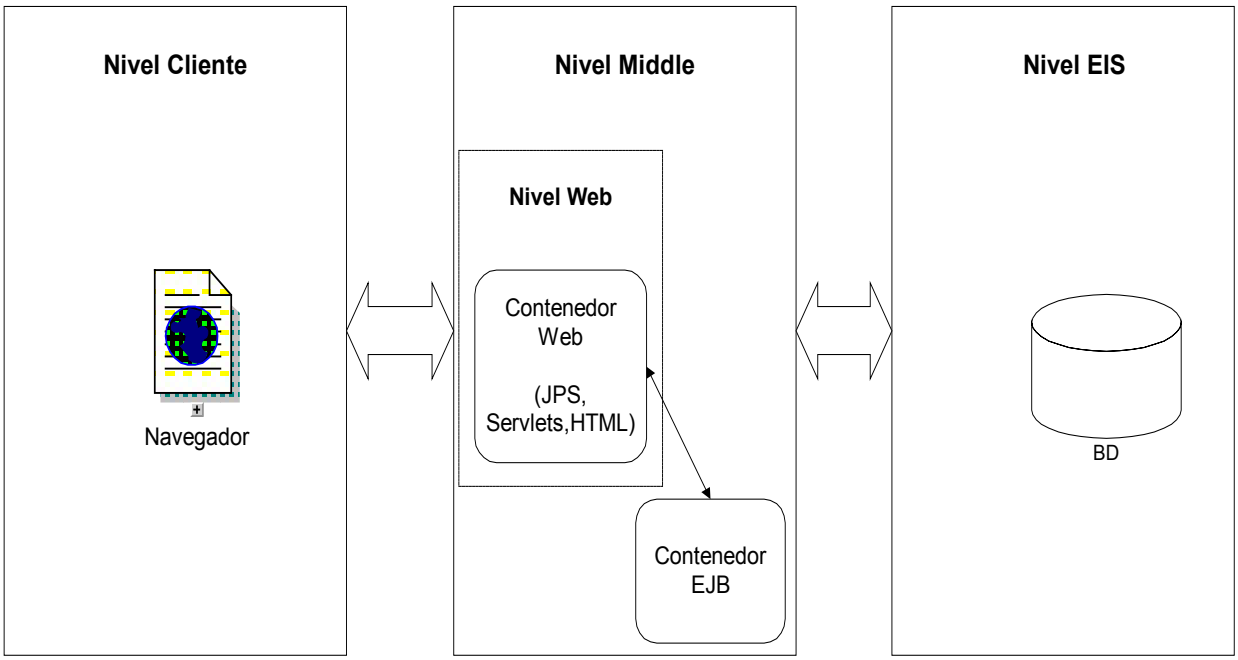


Figura 6.1 - Arquitectura Web

- *Nivel del cliente* : se refiere a la interacción del usuario con la aplicación. La interacción consta de enviar un requerimiento y recibir algún tipo de respuesta del nivel Middle. La interacción puede ser a través de un browser web o también a través de una interfase de servicios web.

- *Nivel de Web*: provee la habilidad para que el nivel del cliente se comunique e interactúe con la lógica de la aplicación que reside en otro nivel. (en muchas aplicaciones Web tradicionales la lógica reside en el mismo nivel). El nivel actúa como un translator y mapea requerimientos http en invocaciones de servicio sobre el nivel Middle.
  Este nivel también es responsable de manejar el flujo de pantalla basado sobre el estado de la aplicación y del usuario.
  Los componentes que residen en este nivel permiten a los desarrolladores extender la funcionalidad básica de un servicio web.

- *Nivel de Middle*: se lo conoce también como "nivel de aplicación" o "server".
  Aplicaciones construidas con este nivel, proveen una arquitectura más escalable, tolerante a fallas y altamente disponible.
  Una de las principales razones de usar un nivel de aplicación es separar las responsabilidades de presentación, de las reglas de negocio y del modelo de la aplicación.

- *Nivel EIS (Enterprise Information System):* contiene los datos y servicios que son usados. Provee accesos a los recursos como BD, mainframes,etc. el middle se comunica con el EIS usando determinados protocolos.

6.1.1 Arquitecturas – Modelo 1, Modelo 2 [Lima00],[J2EE02]

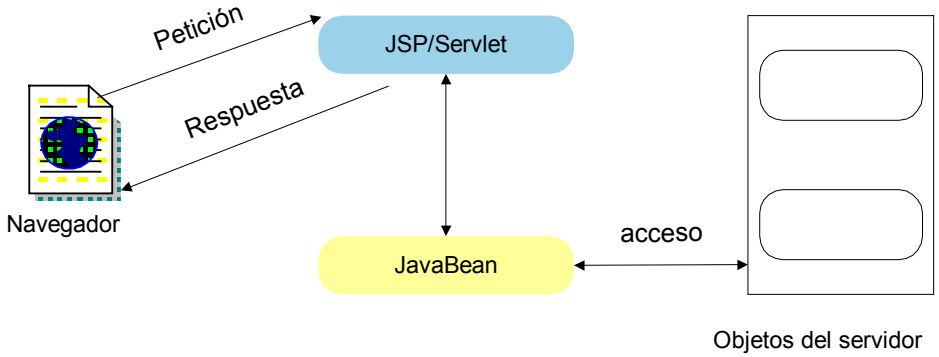
Historia y actualidad

En los últimos tiempos Java se convirtió en el lenguaje más popular para la creación de aplicaciones de servidor. Sun incluso definió una plataforma específica para el desarrollo de aplicaciones distribuidas, J2EE.

Hace un tiempo, el desarrollo de una aplicación Web consistía de una página HTML y un browser, sin datos dinámicos ni aplicaciones de negocios. Esta simplicidad duró poco tiempo, en estos días nos enfrentamos con desarrollo de aplicaciones Web complicadas, incluyendo muchas tecnologías.

Las aplicaciones Web se convirtieron rápidamente en "centradas en JSP", pero sólo utilizar JSP no cubría todas las necesidades de una aplicación Web compleja, entonces las JSP y las servlets se usan juntas. Las servlets podrían ayudar con el control de flujo y las JSPs podrían usarse para la presentación al usuario.

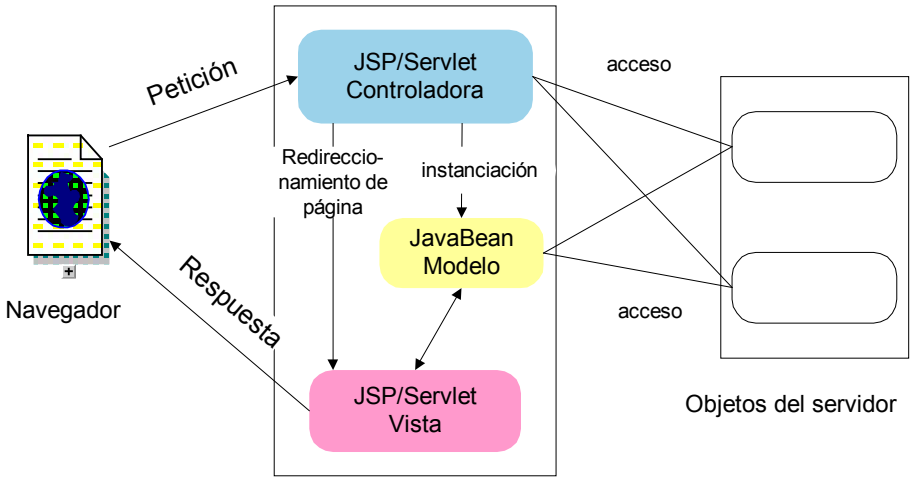
Las especificaciones JSP de Sun abordan la creación de dos modelos distintos para la construcción de aplicaciones usando JSP. La diferencia entre ambos modelos reside en la localización de quién procesa las peticiones de los clientes, es decir, dónde reside la lógica de negocios. Ambos modelos son mostrados en las siguientes figuras



Modelo 1

Figura 6.2

En el modelo 1 el requerimiento del cliente es enviado directamente a la página JSP, ésta se comunica con JavaBeans u otros servicios que representan el modelo de la aplicación, por último determina cuál es la próxima página JSP que verá el cliente. La siguiente vista está determinada por la JSP seleccionada o por parámetros dentro del requerimiento del cliente. En este modelo el control de la aplicación no está centralizado, debido a que la página corriente que está siendo mostrada es la que determina la siguiente página a mostrar, es decir cada página JSP o servlet procesa sus propias entradas (parámetros desde get o post).



Modelo 2

Figura 6.3

En el modelo 2 se introduce una servlet controladora entre el cliente y la página JSP. El cliente no envía directamente el requerimiento a la página JSP (vista), el requerimiento del cliente es primero interceptado por una "servlet controladora". Esta servlet maneja el procesamiento

inicial del requerimiento y determina cuál es la siguiente página JSP que se mostrará al cliente. La servlet controladora centraliza la lógica para enviar los requerimientos a la siguiente vista basada sobre la url de request, parámetros de entrada y el estado de la aplicación. Aplicaciones modelo 2 son más fáciles de mantener y extender dado que las vistas no están relacionadas entre sí directamente.

Comparando ambos modelos

Modelo 1 y Modelo 2 simplemente se refieren a la ausencia o presencia , respectivamente, de una servlet controladora que despacha los requerimientos desde el nivel de los clientes y selecciona las vistas.

La principal diferencia entre los dos modelos es que el Modelo 2 introduce una servlet controladora que es un único punto de entrada y promueve más reuso y extensibilidad que el Modelo 1.

Dentro del Modelo 2 hay una clara separación entre lógica de negocio, vista y quién procesa el requerimiento.

Las aplicaciones Web construidas usando el Modelo 2 son generalmente fáciles de mantener y pueden ser más extensibles que las construidas con el Modelo 1.

El Modelo 1 es bueno para pequeñas aplicaciones con navegación simple y fija y cuando una estructura simple de directorios puede representar la estructura de las páginas de la aplicación.

El Modelo 2 es para aplicaciones Web más complejas.

6.1.2 MVC

El Modelo 2 de JSP separa claramente las responsabilidades en una aplicación Web construida usando las tecnologías Servlet y JSP; es claro que tener una aplicación construida con responsabilidades claras y distribuidas hace que el desarrollo y mantenimiento de la aplicación pueda ser hecho de manera más eficiente.

Un framework de aplicación de MVC puede en gran medida simplificar la implementación de una aplicación modelo 2.

El paradigma MVC divide a las aplicaciones en tres partes:

- Modelo
- Vista
- Controlador

El *modelo* representa la capa de lógica del negocio comentada en el punto anterior.

El *controlador* es el encargado de redirigir o asignar una aplicación (modelo) a cada petición.

La *vista* representa la capa de presentación (Interfase de usuario comentada en el punto anterior).

El funcionamiento es el siguiente: el *controlador* recibe una orden y decide quién la lleva a cabo en el *modelo*. Una vez que el *modelo* (la lógica del negocio) termina sus operaciones devuelve el flujo al controlador y éste envía el resultado a la capa de presentación (*vista*).

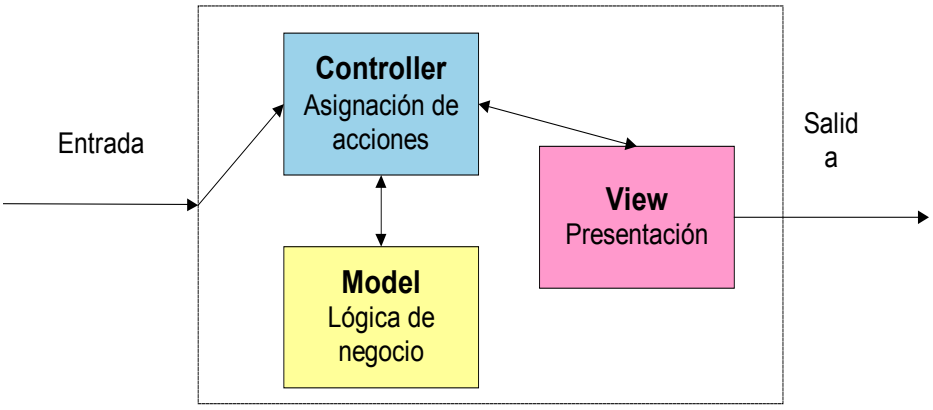


Figura 6.4 - MVC

Normalmente con el patrón MVC existe una notificación de eventos que notifica a la vista cuando alguna porción del modelo cambia.

Sin embargo, dado que un browser en una aplicación Web tiene una conexión sin estado, la notificación del modelo a la vista no puede ocurrir fácilmente. Un usuario por ejemplo puede cerrar el browser en cualquier momento y no se le notifica nada al server. Con aplicaciones Web estándar el cliente debe ejecutar algún requerimiento de vuelta al server para enterarse de los cambios ocurridos en el modelo. Por ejemplo si un usuario está viendo una página con precios de algún ítem y en el mismo momento el administrador cambia alguno de esos precios el usuario no se enterará del cambio a menos que refresque su página.

Ventajas del MVC

Se logra una separación total entre lógica del negocio y presentación, permitiendo que sean modificados independientemente.

La separación de capas, presentación, lógica de negocio y acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenibles.

El mantener los objetos del negocio separados de la presentación hace que la aplicación no esté atada a ninguna presentación en particular. Es muy común que el look-and-feel de los sitios web cambie frecuentemente. Estudios muestran que la frescura de la apariencia de los sitios web ayuda a atraer nuevos visitantes y a que los visitantes existentes vuelvan.

El patrón de diseño MVC provee múltiples beneficios de diseño. Separa lo que concierne al diseño (persistencia de datos y

comportamiento, presentación y control), decrementa la duplicación de código, centraliza el control y hace que la aplicación sea más fácilmente mantenible.

Un framework de aplicación MVC puede facilitar enormemente la implementación de un modelo 2 de aplicación.

Separando la lógica de negocio de la presentación

Ubicar el código de presentación y el de lógica de negocio en lugares separados es una buena técnica de diseño. El nivel de negocio provee sólo funcionalidad de la aplicación, sin hacer referencia a la presentación. El nivel de presentación presenta los datos al usuario delegando la funcionalidad de la aplicación al nivel de negocio.

Esta separación tiene varios beneficios:

- *Minimiza el impacto del cambio*: las reglas de negocio pueden cambiar sin tener que afectar al nivel de presentación y viceversa.

- *Incrementa el mantenimiento*: mucha de la lógica de negocio ocurre en más de un caso de uso de la aplicación. La lógica de negocio debe ser copiada en cada componente que exprese la misma regla de negocio en varios lugares en la aplicación. Futuros cambios a la regla requieren varias ediciones en lugar de una. La lógica de negocio expresada en un componente separado y accedida por medio de una referencia puede ser modificada en sólo un lugar del código origen haciendo que los cambios ocurran en cualquier lugar donde el componente es usado. Beneficios similares pueden ser logrados reusando la lógica de presentación usando stylesheets, custom tags.

- *Provee independencia de cliente y reuso del código*: mezclando la presentación de los datos con la lógica de negocio liga a la aplicación con un tipo de cliente en particular. La lógica de negocio que está disponible referencialmente a través de la llamada a un método sobre un objeto de negocio puede ser usada por diferentes tipos de clientes.

- *Separa roles de desarrollador*: el código que mezcla la lógica de presentación, las reglas de negocio, el procesamiento de los requerimientos hace difícil la lectura para el desarrollador incluso porque hay desarrolladores que están especializados sólo en algunas áreas. Separar la lógica de negocio de la presentación permite que el desarrollador se centre en el área de su experiencia.

6.2 Tecnologías

Para desarrollar el prototipo InCoWeb utilizamos distintas tecnologías Java.

Por un lado utilizamos el framework Struts [O'Re02] para separar en capas a la aplicación y facilitar el desarrollo y posterior mantenimiento. Con Struts nos despreocupamos de la capa del controller y nos enfocamos en el modelo y la vista de nuestra aplicación.

Para las vistas utilizamos la tecnología JSP y los tag-libs, que nos permiten crear contenido dinámico en la web, junto con algunas validaciones y funciones javascript.

El modelo está desarrollado íntegramente en Java.

La persistencia fue realizada con una implementación de la especificación de JDO. [Cra]

Todas las tecnologías mencionadas serán brevemente explicadas a continuación con el fin de entender bien cómo se desarrolló el prototipo.

6.2.1 Java (JSP/Servlets)

Las Servlets son aplicaciones java que se ejecutan dentro de un Web Server. Estas aplicaciones se quedan esperando peticiones de los clientes. Cuando un cliente realiza una petición la servlet genera el código necesario y retorna la respuesta de la petición al cliente. [ApiServ].

La tecnología Java Server Pages (JSP) es la que nos permite crear un contenido dinámico dentro de la Web. Se basa en la tecnología de servlets. Las páginas JSP son páginas que contienen fragmentos de código en Java y código en HTML o XML. [JSP].

6.2.2 Struts [O'Re02] [Struts] [Gul02]

Struts es un framework para aplicaciones Web que implementa el MVC.

Implementa los conceptos del patrón de diseño MVC mapeándolos dentro de componentes y conceptos de una aplicación Web. Provee un conjunto de tag-libs y clases que conforman el *controlador*, la integración con el *modelo* y facilitan la construcción de *vistas*. Struts es una plataforma sobre la que montamos la lógica del negocio y esta plataforma nos permite dividir la lógica del negocio de la presentación entre otras cosas.

El framework fue creado para facilitarles a los desarrolladores construir aplicaciones Web con las tecnologías de Servlets y JavaServer Pages. Al construir una aplicación debe hacerse desde una base sólida a partir de la cual el resto de la estructura pueda crecer según necesite, este principio también tiene que tenerse en mente al construir una aplicación Web.

Al utilizar Struts como base los desarrolladores se centran en aspectos de la construcción de la lógica de negocio de la aplicación.

El framework Struts fue originalmente creado por Craig R. McClanahan y agregado al *Apache Software Foundation* (ASF) en 2000.

Struts nos provee de un framework para implementar un diseño MVC en poco tiempo.

Podemos aplicar Struts a cualquier aplicación Web independientemente de su funcionalidad y cuestiones de rendimiento o arquitectura.

Teniendo en mente la figura 6.1 previamente mostrada de una arquitectura web, ubicamos al framework Struts en el nivel Web. Las aplicaciones Struts son mantenidas por un contenedor web y pueden hacer uso de los servicios provistos por el contenedor, como manejo de request vía protocolo HTTP. Los desarrolladores están libres para centrarse en construir aplicaciones que resuelvan los problemas del negocio.

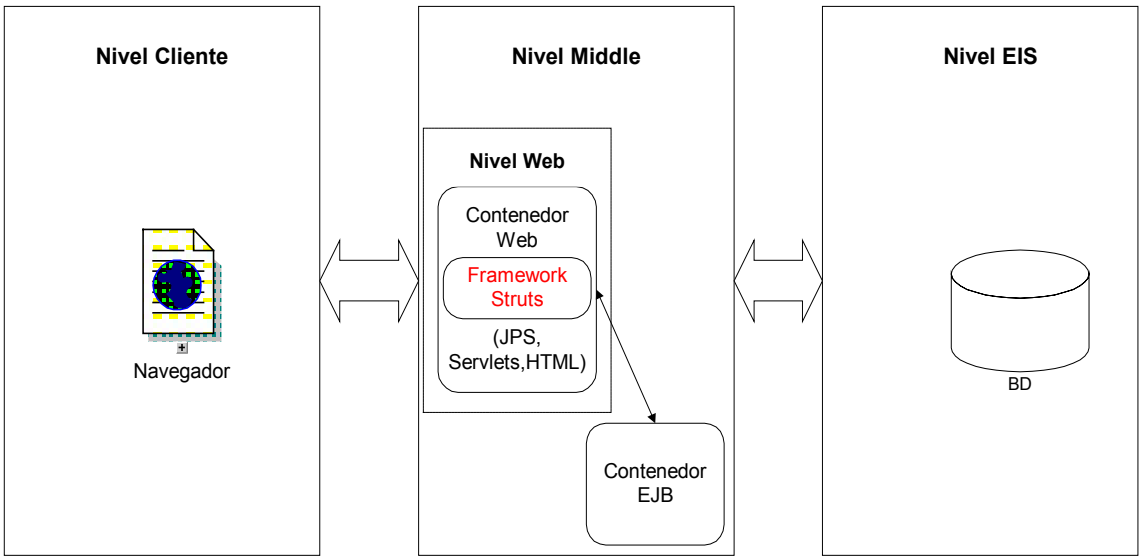


Figura 6.5

Componentes de Struts

El framework puede dividirse en cuatro áreas, tres de las cuales corresponden al MVC; éstas son:

- El *modelo*: representa los objetos que modelan la lógica de negocio de la aplicación.
- La *vista*: formada por un conjunto de páginas JSP, trabajan junto con la servlet controladora y permiten la rápida creación de formularios para una aplicación.
- El *controlador*: es una servlet que despacha los requerimientos de entrada a las clases Action apropiadas.
- Clases útiles para soportar XML parsing, población automática de propiedades de JavaBeans, e internacionalización de prompts y

mensajes.

A continuación detallaremos las diferentes áreas.

Controlador

Como vimos en el MVC el controlador tiene varias responsabilidades, entre ellas están, recibir la entrada del cliente, invocar una operación de negocio y coordinar la vista que se retorna al cliente.

También vimos que con el modelo 2 de arquitectura JSP, sobre el cual Struts fue diseñado, el controlador fue implementado por una servlet Java. Esta servlet se vuelve el punto central del control de la aplicación. La servlet controladora mapea las acciones del usuario dentro de operaciones de negocio y ayuda a seleccionar la vista a retornar al cliente basado sobre el requerimiento y otra información de estado.

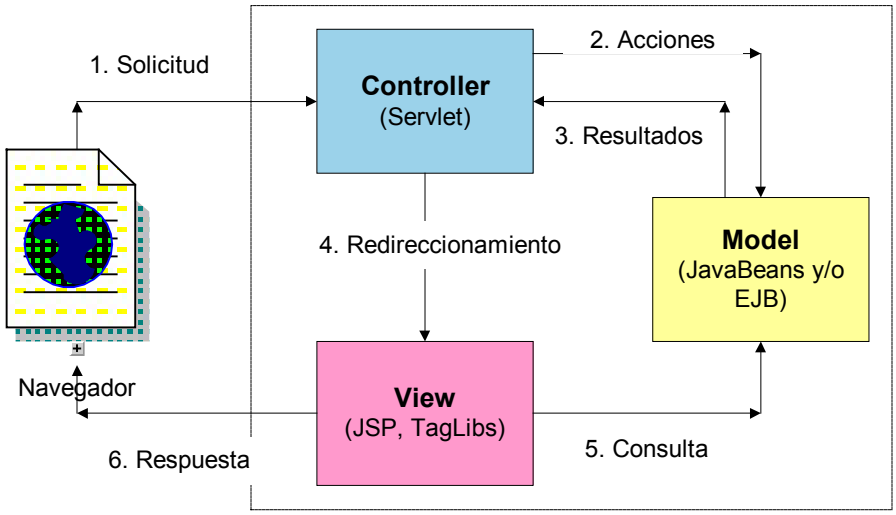


Figura 6.6 – Struts

En el framework Struts, las responsabilidades del controlador son implementadas por diferentes componentes, uno de los cuales es una instancia de *org.apache.struts.action.ActionServlet*.

Clase ActionServlet

La clase *ActionServlet* extiende la clase *javax.servlet.http.HttpServlet* y es responsable de empacar y rutear tráfico Http al manejador apropiado en el framework.

Una vez que el controlador recibe el requerimiento del cliente, delega el manejo del request a una clase helper. Esta clase helper conoce cómo ejecutar la operación de negocio asociada con la acción requerida. En Struts esta clase helper es descendiente de la clase *org.apache.struts.action.Action*.

Clases *Action*

Las clases *Action* actúan como un Adapter entre la acción del usuario y la operación de negocio. La clase Action desacopla el request del cliente del modelo de negocio. Esto permite más de un mapeo uno a uno entre el requerimiento del usuario y una clase *Action*.

El método *execute* de la clase *Action* es llamado por el controlador cuando recibe un request del cliente. Se crea una única instancia de cada clase *Action* en la aplicación. Dado que hay una única instancia de cada clase *Action* en la aplicación, se asegura que todas las clases *Action* operan apropiadamente en un ambiente multi-threaded.

La signatura del método *execute* de la clase Action es el siguiente:

public ActionForward execute
(ActionMapping mapping, ActionForm, form, HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException;

El objeto retornado de tipo *ActionForward* determina el destino para el cual el controlador debe enviar el control una vez que un *Action* fue completado.

Dentro del método *execute* se pueden obtener los datos enviados por el cliente a partir del parámetro form de tipo *ActionForm*, para luego invocar la operación de negocio apropiada.

Hay varias formas de organizar las clases *Action* ha realizar por el desarrollador. Una es que se cree una clase Action distinta para cada acción que el usuario pueda ejecutar y fue la manera que utilizamos en el prototipo.

Mapeos de acción

El controlador usa además de la información de request, un conjunto de mapeos de acción para saber cuál instancia de *Action* invocar cuando recibe un request.

Los mapeos de acción son parte de la información de configuración de Struts que se configura en un archivo XML especial (struts-config.xml). Esta información de configuración es cargada en memoria al inicio y queda disponible al framework en ejecución.

Modelo

Es importante mantener los objetos de negocio separados de la presentación, así la aplicación no queda ligada a un tipo de presentación y pueden modificarse independientemente. Es muy común que el look and feel de un sitio web cambie continuamente. Estudios realizados mostraron que la frescura de la apariencia de los sitios web ayudan a atraer nuevos clientes y a mantener a los existentes.

Struts aconseja mantener la lógica de negocio fuera de las clases *Action* para protegerse de los cambios.

Los objetos del modelo generalmente son específicos de la aplicación, debido a que contienen la lógica de negocio de la aplicación.

Puede existir una correspondencia una a uno entre objetos del modelo y requerimientos de página, aunque también es posible reusar un objeto del modelo para responder a múltiples requerimientos de páginas. Si no hay una acción necesaria para un objeto del modelo, el controlador puede enviar el requerimiento directamente al objeto vista, según esté especificado en el archivo struts-config.xml.

Vista

Los componentes más empleados en Struts para la vista son:

- JavaServer Pages
- Custom tags
- HTML
- Java manojo de recursos (Resource bundles)
- Struts ActionForms
- Value Objects

Clase ActionForm

Los objetos *ActionForm* de Struts son usados en el framework para pasar los datos de entrada del cliente hacia delante y hacia atrás entre el nivel de usuario y el nivel de negocio. Para mantener el nivel de presentación desacoplado del nivel de negocio, no se debería pasar el formulario de acción al nivel de negocio; en lugar de eso; se crea un objeto de valor apropiado usando los datos del formulario y se pasan estos objetos como argumento al nivel de negocio.

Para cada página HTML donde exista un formulario de datos se debería usar un *ActionForm*.

El uso de *ActionForm* también debe registrarse en el archivo de configuración de Struts.

La clase *ActionForm* tiene varios métodos, uno de los más importantes es el método *validate* cuya signatura es:

```
public ActionErrors validate (ActionMapping mapping,
                              HttpServletRequest request)
```

Este método es llamado por el controlador luego de que los valores del requerimiento han sido insertados dentro del bean de formulario.

El bean de formulario puede ejecutar cualquier validación de campos y retornar cualquier error al controlador.

La validación de la lógica de negocio debe ser ejecutada en los objetos de negocio, no en los *ActionForm*.

La validación dentro de los *ActionForm* es acerca de la presentación (por ejemplo que un valor de un field no sea null o vacío). En la nueva versión de Struts 1.1 se agregó un framework llamado Validator para hacer este tipo de validaciones de manera declarativa en vez de programática. Esto hace que el mantenimiento sea más fácil, debido a que las validaciones de presentación se declaran en un archivo xml, de manera que un cambio en dichas validaciones genera sólo un cambio en dicho archivo, sin tener que modificar el código de ninguna clase. Además permite que cada validación pueda ser ligada con diferentes clases *ActionForm* evitando tener que duplicar el mismo código en más de una clase si se hiciera programaticamente.

Los *ActionForm* también son registrados en el *struts-config.xml,* en donde debe indicarse cuáles mapeos de acción usan cuáles *ActionForm.*

Manojo de recursos

Struts al igual que Java provee un conjunto de clases para soportar la lectura de recursos de mensajes.

Con una aplicación Struts se debe proveer un manojo de mensajes para cada lenguaje que se desea soportar.

Desde las páginas JSP se pueden recuperar valores del manojo de recursos utilizando tags especiales para dicho fin.

Un manojo de recursos es usado no sólo para localización. También puede ayudar a reducir tiempo de mantenimiento. Si se usan los mismos mensajes de texto o labels en todo el sitio web o la aplicación; cuando uno o más de estos valores necesitan ser cambiados, sólo hay que hacer el cambio en un único lugar.

Orden de eventos en un procesamiento Struts

- *Web.xml* : una de las principales claves de Struts es el archivo de configuración web.xml. La primera cosa que ocurre, ocurre cuando se levanta el contenedor de JSP. El contenedor de JSP chequeará el archivo de configuración web.xml y determinará qué servlets de acción Struts existen.

- *Un requerimiento* : el segundo paso ocurre cuando un usuario requiere una página o submite un formulario. Este requerimiento es interceptado por el controlador.

- *El controlador*: determina qué acción es requerida y envía la información a ser procesada por un bean de acción. La ventaja principal de tener un controlador es la habilidad de controlar el flujo lógico a través de puntos centralizados, bien controlados.

- *Struts-config.xml* : este archivo almacena los mapeos de acción, esto evita que deba hard codearse el módulo que será llamado

dentro de un componente.
El controlador chequea el archivo struts-config.xml para determinar qué acción invocar. Sólo lee el archivo una vez al inicio, luego accede a memoria donde se almacenó una tabla con los mapeos de acción, para mejorar la performance.

- *El modelo* : se accede a los objetos del modelo para llevar a cabo la lógica de negocio. Sin embargo hay casos en los que no se necesita acceder a objetos del modelo y el controlador puede enviar directamente el requerimiento al objeto vista.

- *La vista*

- *Archivo de propiedades* : es una forma de almacenar mensajes que pueden ser usados por un objeto o página. La ventaja es que permite crear diferentes archivos de propiedades para manejar diferentes lenguajes.

- *La respuesta*: es la página final que ve el usuario

Struts nos provee de un framework para implementar un diseño MVC en poco tiempo.

Podemos aplicar Struts a cualquier aplicación Web independientemente de su funcionalidad y cuestiones de rendimiento o arquitectura.

6.2.3 JDO – Java Data Objects [Cra]

JDO es una especificación de Sun que establece un estándar para persistencia transparente de objetos java en un almacenamiento transaccional.

Dato persistente se refiere a información que sobrevive más allá de los límites de la aplicación que lo creó. La mayoría de las aplicaciones utilizan datos persistentes.

Persistencia transparente es el almacenamiento y recuperación de datos persistentes con poco esfuerzo del desarrollador.

En lenguajes OO como Java los datos manipulados son objetos, pero los almacenamientos de datos tradicionales usados para mantener la persistencia no tienen tal representación. En una base de datos, los datos generalmente son representados por tablas y las relaciones entre ellos son a través de filas y columnas de dichas tablas. Las semánticas de las operaciones de base de datos como create, delete, update no se corresponden con el ciclo de vida de los objetos de nivel de aplicación.

JDO automatiza el mapeo entre objetos de datos persistentes e información en almacenamiento de datos. Evita que el desarrollador

tenga que crear explícitamente los mapeos, permitiéndole centrar su atención en la lógica de negocio de la aplicación.

Los principales objetivos de JDO son :

- Proveer transparencia de la información persistente
- Permitir implementaciones *pluggables* de almacenamientos de datos dentro de servidores de aplicaciones.

JDO provee la ilusión de que la red de objetos recorridos por la aplicación residen todos en memoria, cuando en realidad ellos son activados a medida que lo necesita la aplicación.

Modelo de objetos JDO

En una aplicación típica, las clases de la aplicación están altamente interconectadas.

La aplicación trata con pocas instancias persistentes en un momento determinado y es función de JDO permitir la ilusión que la aplicación puede acceder al grafo entero de instancias conectadas, cuando en realidad sólo un subconjunto de estas instancias necesita ser instanciado en la JVM. Esto es persistencia transparente.

El modelo de objetos JDO de una aplicación está determinado por un conjunto de clases Java y un archivo XML. Cada clase de la aplicación que debe ser persistente debe estar listada en el archivo XML.

Los propósitos principales del archivo XML son:

- Identificar clases persistent-capable y los campos persistentes de dichas clases.
- Especificar las relaciones entre las clases.

Una clase se dice *persistent-capable* cuando sus objetos son persistidos en el almacenamiento de datos. Para que una clase sea *persistent-capable* debe ser aumentada (enhance). Estos aumentos son necesarios para que la clase soporte persistencia transparente. El aumento es el proceso de modificar el byte code de clases ordinarias Java para hacerlas persistent-capable. El enfoque más comúnmente usado para hacer el aumento es leer los archivos java .class y el archivo XML y generar un conjunto de archivos .class aumentados. Uno de los cambios hechos a los archivos .class originales es hacer que la clase implemente la interfase *javax.jdo.PersistenceCapable*.

La implementación de referencia provee un Enhancer de referencia que produce archivos .class aumentados según la especificación JDO. Todas las implementaciones JDO deben soportar clases aumentadas con

dicho Enhancer. Si bien cada proveedor de JDO puede implementar su propio enhancer, éste no puede entrar en conflicto con el Enhancer de referencia. De esta forma las clases aumentadas serán compatibles con todas las implementaciones JDO.

Las instancias de las clases comienzan siendo transient (instancias que existen dentro del contexto de la JVM que las creó) y algunas se convierten en instancias persistentes ya sea de manera explícita con llamadas al método *makePersistence* de una interfase de JDO, o de manera implícita vía persistencia por alcance.

La persistencia por alcance hace una instancia transient , persistente, si ésta es persistent-capable y referenciada por una instancia persistente cuando se hace el commit de la transacción. Esto permite que una gran colección de instancias relacionadas sean persistentes teniendo una instancia de un objeto "raíz" persistida o referenciada por otra instancia persistente.

En particular en nuestro prototipo utilizamos la implementación de JDO, FastObjects j1. FastObjects utiliza un almacenamiento de datos orientado a objetos.

6.3 Arquitectura del prototipo InCoWeb

La siguiente figura muestra un gráfico en donde se integran todas las tecnologías anteriormente mencionadas.

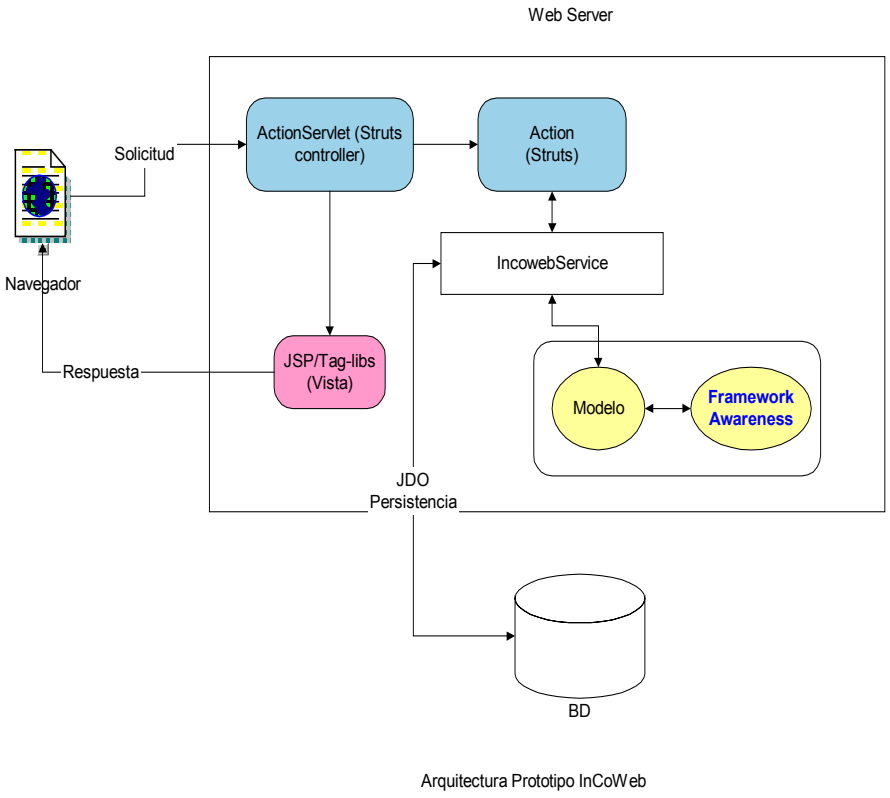


Figura 6.7

Los requerimientos del cliente son manejados por el ActionServlet de Struts, el cual determina qué Action particular debe ejecutarse. Cada Action luego recupera una instancia del IncowebService que es quien se comunica con el modelo y con el framework ACH para registrar todos los eventos de awareness que ocurran en la aplicación, y luego recuperar la información de awareness que debe mostrarse al cliente.

La persistencia de los objetos está manejada por JDO, para la cual utilizamos la implementación de FastObjects.

Las respuestas son devueltas al cliente y visualizadas mediante JSP.

Comentarios finales

En este apartado resumimos los resultados obtenidos, para que el lector tenga una visión general de los aspectos que abarca esta tesis.

Conclusiones

El objetivo principal de este trabajo de grado fue presentar un modelo orientado a objetos de awareness que pueda ser utilizado en el diseño de aquellas hipermedias que tengan características de colaboración y trabajo en grupo.

Comenzamos definiendo al modelo de awareness como una posible extensión de CHDM para manejar la información de awareness. En ese modelo aparecieron 2 clases principales, el AwarenessManager, encargado de la recolección y almacenamiento de la información de awareness y la clase AwarenessComponent para representar los componentes de awareness de cada nodo colaborativo que desee brindar información de awareness.

Para poder mostrar la utilidad del awareness en una aplicación colaborativa diseñamos e implementamos el framework, al que llamamos ACH, este framework define e implementa la clase AwarenessManager del modelo de awareness y además define e implementa 2 jerarquías: la de componentes de awareness y la de eventos de awareness. También definimos 2 interfases para lograr la comunicación entre las clases del modelo de la aplicación y del framework.

Para mostrar el funcionamiento del framework empleamos diagramas de interacción en diferentes escenarios colaborativos.

En el capítulo 5 implementamos un prototipo utilizando todo lo definido. Mostramos los diferentes modelos (conceptual, navegacional, colaborativo y de awareness), para la implementación del prototipo usamos el framework ACH.

Por último en este capítulo explicamos qué cambios deberían ser necesarios si se necesitara extender una aplicación diseñada según nuestra propuesta, y qué cambios deberían hacerse al framework ACH si fueran necesarios otros tipos de awareness.

Podemos concluir que con el modelo de awareness propuesto, se puede a nivel de diseño especificar el tipo de awareness que cada nodo colaborativo de una aplicación va a tener. Es importante remarcar que si es necesario realizar un cambio a nivel de componente de awareness, sólo se necesita agregar una nueva clase en el modelo.

A nivel de implementación un cambio de este tipo, sólo implicaría que el componente compuesto de awareness del nodo colaborativo contenga un nuevo tipo de componente de awareness.

Utilizando el framework ACH un desarrollador se desliga de la responsabilidad de implementar el mecanismo de almacenamiento y

recolección de la información de awareness cada vez que ocurra un cambio que afecte a dicha información.

Como mencionamos en el capítulo 5 es relativamente sencillo realizar un cambio en el framework por requerimientos nuevos. Esto sería necesario sólo si una determinada aplicación necesita registrar información de awareness que no esté contemplada en el framework propuesto.

Nuestro principal objetivo fue brindar no sólo un modelo OO que ayude a los desarrolladores en la etapa de diseño, sino también un soporte en la implementación mediante el framework ACH. Solamente es necesario que la aplicación utilice el framework adecuadamente, instanciando correctamente las clases e implementando las interfases java requeridas del mismo.

Una vez que se aprende a usar ACH, podemos obtener de una manera sencilla una hipermedia colaborativa con capacidades para manejar información de awareness

Por último podemos mencionar que a medida que fuimos avanzando en el desarrollo de este trabajo de grado nos enfrentamos con que ni la metodología OOHDM, ni la extensión propuesta CHDM contemplan la posibilidad de mostrar a cierta información de awareness como un nodo navegacional, este puede ser un punto a explorar en un futuro.

Bibliografía

[ACM01] *"El desarrollo del framework Orientado a objetos"*. ACM Crossroads Student Magazine.2001.

[AO] Anderson, K.; Olof Bouvin,N.*; "Supporting Project Awareness on the WWW with iScent Framework"*. Department of Computer Science, University of Colorado, Boulder, University of Aarhus.

[ApiServ]*Java Servlet* Specification.
http://java.sun.com/products/servlet/index.html

[BZSS] Bibbó, L., Zapico, J., Schümmer, J., Schuckmann, C. *"Collaborative Hypermedia Design Patterns in OOHDM"*.

[ChSMD] Cheverst, K., Smith, G., Mitchell, K. and Davies, N. "*Exploiting Context to Support Social Awareness and Social Navigation"*. Distributed Multimedia Research Group. Department of Computing, Lancaster University.

[CJMS] Cohen, D.; Jacovi, M.; Maarek, Y.; Soroka, V.; *"Collection Awareness on the Web via Livemaps"*. IBM Research Lab in Haifa, Israel.

[Craig] Craig Larman. Applying UML and Patterns: *"An Introduction to Object-Oriented Analysis and Design"*. Prentice Hall.

[Cra] Craig Russell. *Java Data Objects*. JSR version 1.0. http://access1.sun.com/jdo/

[Dix96] Dix, Alan. "*Challenges and Perspectives for Cooperative Work on the Web"*. In Proceedings of the ERCIM workshop on CSCW and the Web. Sankt Augustin, Germany, February 7-9, 1996.

[DB92] Dourish, P. and Bellotti, V. "*Awareness and Coordination in Shared Workspaces"*. In Proceedings of ACM Conference on Computer-Supported Cooperative Work CSCW'92 (Toronto, 1992), 107-114.

[EGR91] Ellis C.A., Gibs S.J., and Rein G.L. "*Groupware: some issues and experiences"*. Communications of the ACM, Vol. 34 not.1, p38-58, Jan. 1991.

[GG] Greenberg, S.; Gutwin, C; *"A framework of Awareness for Small Groups in shared-workspace"*. Department of Computer Science, University of Calgary, University of Saskatchewan, Canada

[GGEM] García, P.; Gomez Skarmeta, A.;Egea, J.; Martínez, E. *"C-CSCW A Corba Approach for Computer Supported Cooperative Work "*.Universidad de Murcia, Facultad de Ciencias de la Computación, Murcia, España.

[GG95] Gutwin, C. and Greenberg, S. "*Support for Group Awareness in Real-Time Desktop Conferences"*. In Proceedings of The Second New Zealand Computer Science Research Students' Conference, University of Waikato, Hamilton, New Zealand, April 18-21, 1995.

[GG96] Gutwin, C. and Greenberg, S. "*Workspace Awareness for Groupware"*. Conference companion of the Conference on Human Factors in Computing Systems (CHI'96), p208-209. Vancouver, 1996.

[Ggree96] Gutwin, C. and Greenberg, S. "*Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation"*. Springer-Verlag, 281-298, 1996.

[GGR96] Gutwin, C., Greenberg, S., and Roseman, M. "*Staying Aware in Groupware Workspaces"*. Proceedings of Video ACM CSCW'96 Conference on Computer Supported Cooperative Work. Boston, USA, ACM Press., 1996.

[GGRose96] Gutwin, C., Greenberg, S., and Roseman, M. "*Supporting Workspace Awareness in Groupware"*. Proceedings of Video ACM CSCW'96 Conference on Computer Supported Cooperative Work, Boston, USA, ACM Press., 1996.

[GHJV94] Gamma, E., Helm, R., Johnson, R., Vlissides, J. "*Design Patterns: Elements of Reusable Object-Oriented Software"*. Addison-Wesley, Massachusetts, 1994.

[Gree] Greenberg, S.; *"Real Time Distributed Collaboration "*. University of Calgary, Canada. www.cpsc.ucalgary.ca/~saul

[GR96] Gutwin, C.; Roseman, M. *"A Usability Study of Workspace Awareness Widgets"*. Department of Computer Science, University of Calgary, Canada.1996

[GR99] Greenberg, S. and Roseman, M. "*Groupware Toolkits for Synchronous Work"*. In M. Beaudouin.Lafon, editor, Computer-Supported Cooperative Work (*Trends in Software 7*), Chapter 6, 135-168, John Wiley & Sons Ltd., 1999.

[GSG] Gutwin, C.; Stark, G.; Greenberg, S.; *"Support for Workspace Awareness in Educational Groupware"*. Department of Computer Science, University of Calgary, Canada.

[Gul02] Sulzar, N. *"Fast Track to Struts. What it Does an How"*. Published on TheServerSide. November 2002.

[GutGR96] Gutwin, C., Greenberg, S., and Roseman, M. "*A Usability Study of Workspace Awareness Widgets in a Shared Workspace Groupware System"*. Proceedings of Video ACM CSCW'96 Conference on Computer Supported Cooperative Work, 1996.

[Gut97] Gutwin, C. "*Workspace Awareness in Real-Time Distributed Groupware"*. PhD Dissertation, Department of Computer Science, University of Calgary, Canada. Available at: www.cpsc.ucalgary.ca/grouplab/papers/. 1997.

[GutGree] Gutwin, C., Greenberg, S.,"*A Framework of Awareness for Small Groups in Shared-Workspace Groupware"*.

[JSP] *JavaServer Pages* Specification.Java Community Process**.** http://java.sun.com/products/jsp

[J2EE02] *"Designing Enterprise Applications with the J2EE™ Platform, Second Edition"*. 2002 Sun Microsystems, Inc. All Rights Reserved.

[Kind96] Kindberg, Tim. "*Mushroom: A framework for collaboration and interaction across the Internet."* . In Proceedings of the ERCIM workshop on CSCW and the Web. Sankt Augustin, Germany, February 7-9, 1996.

[Koch98] Koch Thomas. *"Groupware on the Internet"*. Diploma Thesis. Technical University Graz. Institut für Informationsverarbeitung und Computergestützte neue Medien, 1998.

[KS97] Kirchner, L., Schuckmann, C. "*Groupware Developers Need More Than Replicated Objects"*. In Proceedings of the OOGP-Workshop, European Conference on Computer Supported Cooperative Work 1997 (ECSCW'97), Lancaster, UK, September 7-11, 1997.

[Lima00] *"Manual avanzado de Java"*.Ediciones Anaya Multimedia.2000

[MR01] Mola, V; Russo W.*, "Aplicando Métodos formales a la Construcción de Aplicaciones de Hipermedia Colaborativas"*. Trabajo de grado. LIFIA- Laboratorio de Investigación y Formación en Informática Avanzada. Departamento de Informática, Facultad de Informática, Universidad Nacional de La Plata.

[Moody00] Moody, P.B. "*The Role of Awareness in Social, Collaborative and Shared Activities"*. Position Paper for CSCW 2000. International Workshop on Awareness & the World Wide Web, 2000.

[Nie95] Nielsen, J. "*Multimedia and Hypertext, the Internet and Beyond"*.Academic Press,Cambridge. 1995.

[O'Re02] Jakarta Struts – Mastering Java Web Application with Courage
The O'Reilly Struts book – Struts 1.0 y 1.1
Chuck Cavaness
O'Reilly and Associates, Inc. All rights reserved. *Struts book*. 2002

[PXAI] Implementación del MVC en Aplicaciones Web
http://ciberia.ya.com/pxai/ma.html

[PWWF] Podgorny, M.; Walczack, K.; Warner, D.; Fox, G.; *"Internet Groupware Technologies- Past, Present and Future "*. Northeast Parallel Architectures Center, University of Syracusa.

[RGC96] Rossi, G.; Garrido, A.; Carvalho, S.; *"Design Patterns for Object Oriented Hypermedia Applications"*. LIFIA- Laboratorio de Investigación y Formación en Informática Avanzada. Departamento de Informática, Facultad de Informática, Universidad Nacional de La Plata.

[Ross96] Rossi, Gustavo. "*Un metodo orientado a objeto para projeto de aplicaçoes hipermidia"*. PhD thesis, PUC-Rio, Brasil, julho, 1996.

[SAR] Schwabe, D.; Almeida Pontes; Rita de; Moura, J. *;"OOHDM-WEB: An Environment for Implementation of Hypermedia Applications in the WWW"*.Departamento de Informática, PUC-Rio.

[Sch89] Scheniderman, B. *"Reflection of authoring, editing and managing hypertext"*. In E. Barret editor, the society of text. Cambridge,Mass: MIT Press,1989.

[SCT00] Scott W. Ambler . "*Mapping Objects to Relational Databases*" http://www.AmbySoft.com/mappingObjects.pdf
Octubre, 2000

[SKK] Sakata, Konomi, Kambayashi. "*Environment Awareness Support for Customizable Shared Hypermedia Documents"*. Dept. of Information Science, Faculty of Engineering, Kyoto University. Yoshida-Honmachi, Sakyo, Kyoto 606-01, Japan.

[SKB97] Schlichter, J., Koch, M., Bürger, M.. "*Workspace Awareness for Distributed Teams"*. In Proc. Workshop Coordination Technology For Collaborative Applications, Singapore, 1997, W.Conen (ed.), Lecture Notes on Computer Science, Springer.

[SSSS] Schuckmann, C.; Schümmer, J.; Steiz, P.; Steinmetz, R.; *"Moddeling Collaboration using Shared Objects"*. GMD-German National Research Center for Information Techology. IPSI-Integrated Publication and Information Systems Institute. Darmstadt, Alemania.

[SSS99] Schuckmann C., Schümmer J., Seitz P. "*Modelling Collaboration using Shared Objects"*. GMD – German National Research Center for Information Technology. Integrated Publication and Information Systems Institute – IPSI . Germany.In Proceedings of ACM GROUP'99. 1999.

[Struts] Jakarta Project. Struts. http://jakarta.apache.org/struts.

[TS98] Terzis, Sotiris (1998). "*CSCW & Groupware"*. GMT, 1998.