WorkShop
EIG2009

WORKSHOP INTERNACIONAL
EIG2009
Departamento de Ingeniería de Sistemas
Universidad de La Frontera – Chile
Diciembre 3 y 4, 2009

Utilización de la programación Orientada a Aspectos para Promover la Colaboración Casual en Ambientes Web.

Claudio Gutiérrez-Soto[1], Pedro Campos[1], Angélica Urrutia[2], Luis Guerrero[3]

[1] Universidad del Bío-Bío, Av. Collao 1202, Casilla 5C, Concepción, Chile
[2] Universidad Católica del Maule, Av. San Miguel 3605, Casilla 617, Talca, Chile
[3]Universidad de Chile, Av. Blanco Encalada, Casilla , Santiago, Chile
cogutier@ubiobio.cl, aurrutia@ucm.cl, luguerre@dcc.uchile.cl

Resumen. Hoy en día más personas utilizan la Web como herramienta para interactuar colaborar y realizar trabajo en grupo. Generalmente estas son implementadas en ambientes que proveen programación concurrente. Desafortunadamente al tratar de unir el paradigma orientado a objetos con la programación concurrente se dará pie a la existencia de características implícitas en el sistema resultante que no podemos modelar de manera eficiente con dichos paradigmas. Dichas características o propiedades se ven reflejadas en aspectos tales como: sincronización, y restricciones en tiempo real, por nombrar algunos. Dichas propiedades se denominan aspectos. Uno de los principales aportes de este artículo es una solución de alto nivel a través del uso de patrones de diseño, utilizando el paradigma de la programación orientada a aspectos. Otro aporte, es el desarrollo de una arquitectura Web, a través de la cual se puede proveer las condiciones necesarias para que los usuarios se puedan comunicar y posiblemente interactuar.
Palabras claves: world wide web, percepción de usuario, programación orientada a aspectos.

1. Introducción

Las personas utilizan Internet de diversas formas y para distintos propósitos, tales como: entretenimiento, educación, recuperación de documentos, y comunicación, por mencionar algunos. No obstante, Internet es utilizado principalmente como herramienta de comunicación. Esta comunicación incluye contacto con colegas, amigos, familia y grupos sociales. A través de esta comunicación, Internet y en particular la Web pueden traer no sólo beneficios sociales individuales, sino que también a organizaciones [1], comunidades y a la sociedad en general. Internet cuenta con muchas aplicaciones que promueven la comunicación, reduciendo los costos de esta comunicación entre las personas que se encuentran distribuidas geográficamente. Además, estas personas pueden conformar comunidades virtuales a través de listas de intereses, y sala de conversaciones, por mencionar algunas). Estas comunidades están organizadas típicamente por un tópico específico, área, o actividad en común. Aparentemente Internet, y en particular la Web, no sólo supera la barrera de la distancia geográfica, sino que también supera las barreras idiomáticas y de idiosincrasia de estas comunidades, las cuales pasan a conformar Comunidades Virtuales.

Las personas son invisibles en la Web. Sin embargo, espacios de interacción social ya existen en la Web (como por ejemplo, sala de comunicación, salas de juegos) y algunos sistemas como espacios virtuales [2], generalmente la interacción social en la Web se ve limitada sólo por la lectura y comentario de determinados documentos, no obstante, hoy en día es posible encontrar arquitecturas que promueven la colaboración casual [3], así como también sitios con herramientas [4].

Por otro lado, los ambientes en el mundo real han sido diseñados para la interacción. Reuniones casuales en cafeterías, fotocopiadoras promueven la creatividad espontánea. En estos lugares las personas saben quién está a su alrededor, cómo se contactan, y cuán próximos se encuentran.

Muchas organizaciones están al tanto de los resultados que pueden traer dichos espacios, y por ende dichos espacios son diseñados para promover el trabajo en equipo. Más aún, en estos espacios no es necesario un soporte tecnológico particular.

Sin embargo, en situaciones donde los miembros del grupo se encuentran geográficamente distribuidos, proveer una interacción espontánea es mucho más complicado y la tecnología utilizada debe ser la adecuada para proveer dicha característica.

A partir de la década de los 90's se comenzó a realizar estudios sobre los espacios compartidos [5], [6] y sobre la percepción, en particular sobre la percepción grupal [7], [5]. Aunque existen varias definiciones de percepción, una de las más utilizadas es la que dan Dourish & Belloti, quienes definen percepción como un entendimiento compartido de las actividades de los demás, que proveen un contexto para la propia actividad [5]. Con respecto a los estudios de espacios compartidos, éstos hacían énfasis en la importancia de la espontaneidad, o del comportamiento humano improvisado. La mayoría de estos autores sugirieron que la percepción grupal es la información referente a la presencia y a las actividades de los demás integrantes del grupo.

Por otro lado, en la Web, el interés por atraer a los visitantes es evidente, y generalmente este interés es medido a través de estadísticas que reflejan el éxito de dicho sitio. No obstante, la mayoría de los sitios en la Web carecen de la posibilidad de proveer percepción de usuario (ver a otros usuarios) por lo que la posibilidad de interactuar con posibles pares, y de generar comunidades virtuales, son casi nulas. Esto en contraste con lo que sucede en los espacios reales, donde las personas tienen la posibilidad de percibir la presencia del resto. El propósito principal de este artículo es promover una arquitectura orientada a objeto que promueva percepción, interacción y colaboración de usuarios de una manera eficiente a través del paradigma orientado a aspectos.

Este artículo está organizado de la siguiente manera. En la sección 2, se presenta el estado del arte de la percepción de usuario en la Web, así como también de las comunidades virtuales. En la sección 3, se describe la importancia de la percepción de usuario, en la sección 4 se describe la nueva arquitectura, en la sección 5 se discuten los niveles de comunicación e interacción. Finalmente, en la sección 6 se dan las conclusiones.

2. Comunidades virtuales y percepción de usuario en la Web

Una comunidad virtual es definida como un grupo de personas que comparten intereses en común, donde generalmente estas personas se encuentran distribuidas en tiempo y espacio. Este grupo de personas se encuentran unidas a través de sistemas computacionales. Las personas que usan la Web pueden conformar Comunidades Virtuales porque ellos comparten intereses. Más aún, generalmente los integrantes de estas comunidades virtuales se encuentran geográficamente distribuidos.

Por otro lado, la gran mayoría de las personas que conforman la Web, sienten la necesidad de interactuar con sus pares. Estas actividades de interacción no necesariamente envuelven a los miembros de una comunidad en una tarea o propósito en común, sino que estas actividades pueden involucrar el intercambio de información o de nuevos intereses entre parte de los miembros de la comunidad.

Hoy en día, hay más oportunidades de acceder a Internet, porque existen una gran gama de proveedores de Internet, así como también de los lugares públicos que proveen acceso. Más aún, los costos para acceder a este tipo de servicios han ido disminuyendo de manera considerable en los últimos años. Este acceso masivo a Internet debería de estimular el crecimiento de comunidades virtuales. Estas comunidades virtuales deberían de conformar sitios para promover la interacción de los usuarios en la Web.
Es importante ver como la Web se relaciona con la Percepción. Una primera perspectiva es considerar a la Web simplemente como una implementación que la soporta. En los proyectos NYNEX Portholes [8] o @Work [9], las páginas Web son usadas para desplegar vividas imágenes desde distintos lugares (usualmente el espacio de trabajo de los colegas). En otras palabras, la Web es utilizada para transportar información que refleje las actividades que están ocurriendo en el mundo real, lo cual también es utilizado como una plataforma que implementa percepción.

Otro punto de vista es considerar en sí misma a la Web como un espacio, donde las actividades sociales toman

lugar. Las personas entran y salen, ordenan productos, y responden preguntas, por mencionar algunas actividades. De esto provienen dos preguntas interesantes: primero, cómo nosotros podríamos acercar la noción de percepción de las otras personas en la Web; segundo, cómo podríamos hacer que las personas puedan percibir en línea, qué es lo que ésta ocurriendo en su espacio. La primera pregunta ha sido abarcada por varios sistemas [10]. El Internet Foyer [11] y en algunos aspectos el Dangling String [12] y el Tangible Bits de Ishii [13], están abocados a responder la segunda pregunta.

No obstante, también podemos encontrar otras aproximaciones como WebRogue [4], las cuales hoy proveen percepción de usuario a través de secciones de conversación con el uso de Browser. No obstante, no es posible establecer comunicación con personas fuera de línea.

Uno de los aportes de este artículo es poder proveer información de los usuarios que estuvieron o que están en el mismo sitio Web.

3. Interacción social en la Web a través de la percepción de usuario

La Web es un enorme repositorio de información distribuida, donde las personas pueden acceder a todos los documentos virtuales existentes. Las personas visitan la Web para recuperar esos documentos. Sin embargo, las personas que visitan la Web, tienen diferentes razones, por ejemplo: leer noticias, leer e-mail, y buscar información, por nombrar las más importantes. En Morrison [14] podemos encontrar un Método Taxonómico, en donde la mayoría de las personas están dedicadas a la actividad de búsqueda de información que no necesariamente corresponden a un tópico en particular. Por otro lado, Sellen [15] propone la construcción de un framework, el cual describe las diferentes tareas que se realizan en la Web. Aquí los resultados relevantes se concentran en las actividades de búsqueda, con el fin de suplir tareas futuras (tales como la recolección de información para escribir un documento, o para preparar una reunión); como herramienta inspiradora, o para obtener ideas. Sellen [15] también muestra información estadística referente al tiempo de duración de las diferentes actividades en la Web, donde la recolección de información es una de las más relevantes debido a los periodos de tiempo invertidos en esta tarea. Sin embargo, la mayoría de las personas no tiene tiempo suficiente para explorar todas las fuentes ofrecidas por las máquinas de búsqueda, cuando las personas no conocen un URL específico. Por otro lado, dependiendo del tiempo de las personas, ellos sólo pueden recuperar una parte de la información disponible en la Web. La porción de información recuperada corresponde a un subconjunto de información disponible en la Web. Sin embargo, el subconjunto de información recuperada no necesariamente llena las expectativas de los usuarios. Por otro lado, si las personas pudieran percibir un conocimiento compartido acerca de las fuentes existentes, la percepción podría ser un elemento relevante para alcanzar los objetivos de los usuarios y disminuir el tiempo de búsqueda en la Web.

Los usuarios de la Web navegan a través de páginas, las cuales son creadas independientemente de la localización física de los usuarios. Por otro lado, la interacción a través de los servidores Web es efectuada de manera asincrónica o semi-asincrónica. Sin embargo, es muy difícil para un usuario particular participar en una sesión en la Web, donde las personas negocien y compartan documentos, esto debido principalmente a la falta de percepción de los otros usuarios y de las actividades que éstos realizan.

La falta de percepción entre los usuarios es la principal limitación de los sistemas interactivos basados en Web. No existe una forma simple de saber qué personas están o estuvieron navegando en las mismas páginas.

La arquitectura propuesta provee un tipo de percepción real, donde el usuario final es capaz de ver qué personas están o han visitado las mismas páginas Web. Más aún, esta arquitectura provee posibilidades reales de establecer comunicación con sus posibles pares por medio de una arquitectura eficiente soportada por la Programación Orientada a Aspectos.

En la siguiente sección, se describe el modelo de la arquitectura. Además, se describe cada componente que la conforma, cómo están interrelacionados los componentes y cómo opera la programación orientada a aspectos en dicha arquitectura.

4. Arquitectura que promueve la percepción de usuario en la Web

El objetivo principal del desarrollo de esta nueva arquitectura, es que ésta es capaz de proveer percepción a los usuarios de la Web, acerca de qué usuarios podrían eventualmente compartir los mismos intereses. Esto significa que un usuario que visita una página Web y está interesado en el contenido de ésta, podría registrarse, presentando sus áreas de interés y al mismo tiempo ser visto por otros usuarios que han realizado el mismo procedimiento. Esta característica de ver a otros usuarios que están interesados en los mismos tópicos, nos da la posibilidad de crear una colaboración casual entre ellos.

En esta sección, se presentan las características principales de nuestra arquitectura. Todos los componentes han sido construidos en el lenguaje Java, los aspectos fueron implementados en AspectJ. Primero, se describirá de manera general cómo opera el sistema, cómo los usuarios se registran en el sistema y cómo indican sus intereses, cómo los usuarios reciben información de otros usuarios, cómo los usuarios podrían utilizar esta información para establecer contacto con estas personas, cómo los datos son almacenados en esta arquitectura, finalmente como opera el patrón observador implementado en Aspecto.

4.1 Implementación de los objetos

El modelo para proveer percepción de usuarios en la Web, está compuesto por: Un Browser, un servidor Web, un servidor de Meta-Datos y un Demonio Proxy. La relación entre los diferentes componentes de la arquitectura está ilustrada en la Figura .1.

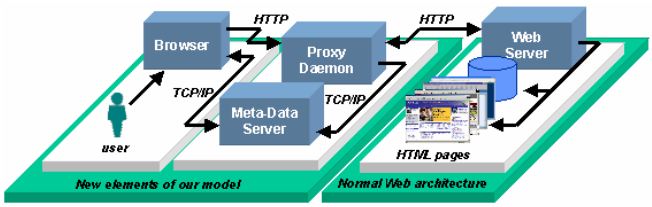

Fig 1. El modelo arquitectónico

Cuando un usuario solicita una página utilizando nuestro Browser, éste envía los requerimientos al Demonio Proxy. El proxy registra alguna información en el servidor de Meta-Datos antes de pasar los requerimientos al servidor Web. Después el servidor Web envía las páginas HTML que el usuario está requiriendo. El servidor de Meta-Datos contiene información de percepción de usuario que debería ser mostrada dependiendo de la página requerida o visitada por el usuario. En las próximas secciones explicamos de manera detallada cada componente de nuestro modelo.

4.1.1 El Browser

Construimos un Browser que opera de igual manera que los browser tradicionales utilizados para la Web. Éste busca y despliega páginas HTML. Sin embargo, el usuario puede entrar a una sección de percepción previa autenticación en los servidores, si él desea ver otros usuarios y si también quiere ser visto. Cada usuario que usa este Browser tiene la posibilidad de ver a otras personas con las mismas inquietudes e intereses en una página Web específica, siempre que ellos también se hayan autenticado.

Los Browsers reciben información acerca de las personas que han visitado la misma página Web. Dicha información es proporcionada por el servidor de Meta-Datos. Por otro lado, nuestro Browser realiza un match entre el URL del sitio y el URL provisto por el usuario, proveyendo feedback acerca de las personas que han visitado la misma página Web. El máximo número de usuarios desplegados por la lista es 15, para a evitar la sobrecarga de información. El criterio utilizado para desplegar información corresponde a las últimas visitas que se le han hecho a la página.

Respecto a la interacción con los otros componentes de la arquitectura, el Browser se comunica directamente con el Demonio Proxy y el servidor de Meta-Datos. En el Demonio Proxy, la interacción es producida por el Browser, el cual hace un requerimiento de algunos documentos Web al servidor Web, pasando primero a través del Demonio Proxy. La interacción entre el Browser y el Demonio Proxy es a través del protocolo HTTP. La interacción entre el Browser y el servidor de Meta-Datos está enfocada en proveer información a los usuarios del Browser. La interacción entre estos componentes es a través del protocolo TCP/IP.

Respecto a la interfaz de usuario, pensamos que es muy importante mantener el estándar de las interfaces de los Browser populares, porque las personas están familiarizadas con su operación. En la Figura 2, mostramos el Browser implementado. Éste está compuesto por tres paneles: en el lado izquierdo, se

muestra los nombres de las personas que ya se han registrado en el sistema y que han visitado esta página recientemente. En el lado derecho, se muestra la página Web actual que el usuario está visitando, en la parte superior existe una pequeña barra de herramientas que provee las funciones básicas que poseen los Browser convencionales.

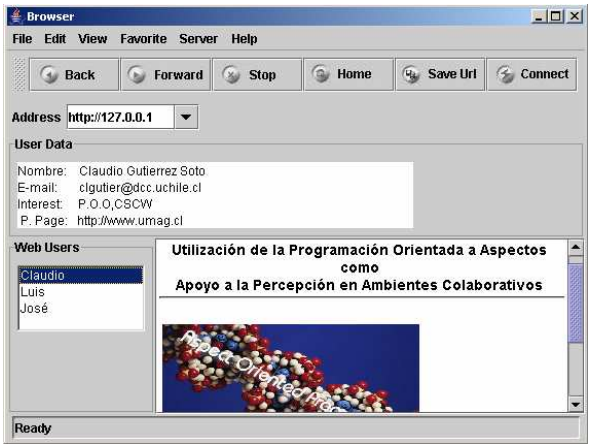

Fig 2. Browser

4.1.2 El servidor de Meta-Datos

El servidor de Meta-Datos, contiene información acerca de los usuarios registrados y acerca de los hechos que han ocurrido en el sitio Web. Esta información puede incluir aspectos como: qué usuarios han sido autentificados, a qué hora y cuándo han sido visitadas las páginas.
Cuando un Browser realiza una solicitud al servidor Web, si el usuario está autentificado, entonces el servidor de Meta-Datos envía información a los usuarios acerca de las otras personas que han visitado la misma página Web. Con la información proporcionada por el servidor de Meta-Datos, el usuario puede decidir comunicarse con otros usuarios que tienen los mismos intereses. Por otro lado, la información que contiene el servidor de Meta-Datos puede ser extendida a otro tipo de información que puede ser muy útil para establecer una mejor comunicación con las otras personas.

La comunicación entre el servidor de Meta-Datos y los Browser, es a través del protocolo TCP/IP. Es importante destacar que la información provista por el servidor de Meta-Datos puede ser asincrónica. Esto se debe principalmente a que los usuarios podrían estar interesados en información como: quiénes y cuándo visitaron la página Web.

4.1.3 El Demonio Proxy

El Proxy es un demonio que recibe llamados desde los Browsers. Estas llamadas son originalmente enviadas al Servidor Web. El Demonio Proxy no ofrece directamente los servicios que el servidor Web promueve, pero invoca los servicios ofrecidos por el servidor Web. El Demonio Proxy está diseñado conforme al patrón proxy propuesto por Gamma [16].
Si el usuario está autentificado, el Demonio Proxy toma estos requerimientos, capturando los datos de interés y almacenando esta información en el servidor de Meta-Datos. Cuando la información es almacenada en el servidor Meta-Datos, se produce una comunicación directa entre el servidor de Meta-Datos y el Browser. Si existe comunicación entre el servidor de Meta-Datos y los Browser, el Demonio Proxy opera de manera normal mientras el usuario no abandone la sección. Si el usuario abandona la sección, el Demonio Proxy abandona sus funciones para dicho usuario. La comunicación entre el servidor de Meta-Datos y el Demonio Proxy es a través del protocolo TCP/IP.

4.1.4 El Servidor Web

El servidor Web que hemos implementado, es un pequeño servidor hecho en Java. Los datos que contiene el Servidor Web pueden ser imágenes, video y voz. Designamos a este tipo de información Documentos Web. Sin embargo, es posible utilizar cualquier servidor Web, por que nuestro modelo no cambia la funcionalidad original de este componente, ni la funcionalidad de las páginas que éste provee (ver figura 1).

4.2 Programación Orientada a Aspectos

En [17] se habla del problema que existe entre la programación concurrente y la programación orientada a objetos. En éste se describe que la programación

orientada a objetos opera de manera eficiente si el problema puede ser descrito con interfaces de relativa simpleza las cuales operan entre los objetos de un sistema. No obstante, en un ambiente distribuido dichas interfaces dejan de ser eficientes debido a la concurrencia, perdiendo todos los beneficios alcanzados por las características de la programación orientada a objetos.
Esto debido principalmente a que existen características implícitas en un sistema que no podemos modelar de manera eficiente, pero que tratamos de abarcarlas sin mucho éxito con los paradigmas de programación tradicionales [18]. Dichas características o propiedades no provienen de manera aleatoria, pero tienden a constituir entidades emergentes, las cuales afloran en tiempo de ejecución. Algunos ejemplos de dichas características o propiedades son: sincronización, programación de tareas, asignación de recursos, optimización de la ejecución, manejo de fallas, persistencia, comunicación, coordinación, administración de memoria y restricciones en tiempo real.
Dichas propiedades son denominadas aspectos, y fueron originalmente introducidas por Kiczales [18]. Los aspectos son definidos como propiedades del sistema que tienden a acortar (cut across) los componentes funcionales, incrementando su interdependencia y afrontando el problema de código complejo desde una nueva perspectiva.
En términos generales podemos decir que un programa orientado a aspectos está compuesto por los programas orientado a objetos, más el programa orientado a aspectos. Aquí se linkea el funcionamiento de los programas orientados a objetos con el de aspectos, donde los aspectos modifican o alteran el comportamiento normal de los objetos. No está demás mencionar, que además del beneficio de la eficiencia (acortar por ejemplo la funcionalidad o invocación de métodos), no es necesario modificar los programas originales orientados a objetos.
AspectJ es un lenguaje de propósito general que extiende Java, desarrollado por el Centro de Investigación de Xerox Palo Alto. Éste habilita las implementaciones plug-and-play a través de los cortes de restricciones sobre código Java. AspectJ posee la misma semántica y sintaxis que Java, pero incorpora un nuevo concepto llamado join point.

Los join point son puntos de ejecución de un programa que están bien definidos. Éstos incluyen métodos, llamadas a constructores y el acceso a atributos. Un aspecto, es un tipo de corte o atajo sobre la funcionalidad de las clases. Desde la noción de Java, es una unidad de código modular, con una interfaz bien definida, que hace posible lograr eficiencia en la ejecución de un programa en tiempo de compilación. Los aspectos son definidos a través de una declaración explícita.
Un pointcut es un conjunto de join points que expone algunos de los valores en el contexto de ejecución de esos join points. Existen varias primitivas de diseño de pointcut, nuevos nombres de pointcut pueden ser definidos a través de la declaración explícita de pointcut. El advice es código que ejecuta cada join point en un pointcut. Los advice tienen acceso a los valores expuestos por los pointcut. Los advice quedan definidos por las declaraciones before, after y around.
Un introduction es código que debería cambiar el tipo de estructura del programa, agregando o extendiendo las interfaces y clases, con nuevos campos, constructores o métodos. Los introduction están definidos a través de una extensión de métodos usuales, campos y declaración de constructores.

4.2.1 El patrón observador

Los patrones de diseño describen un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software. Por lo que entre sus beneficios podemos mencionar que éstos entregan una buena solución, la cual ya ha sido probada, nos ayudan a diseñar correctamente en menos tiempo, nos ayudan a construir soluciones reutilizables y extendibles, facilitan el aprendizaje y facilitan la comunicación entre los miembros del grupo, por mencionar las mas relevantes [16].
Por otro lado, el contexto sobre el cual se desarrolla la percepción de datos, se ambienta al uso de los objetos que conforman nuestra arquitectura. En dicha arquitectura es de suma importancia contar con percepción de datos, ya que nuestra aplicación se encuentra inserta en un ambiente concurrente y distribuido.
Dentro de los patrones de comportamiento, podemos encontrar un patrón que de acuerdo a sus características resulta idóneo para ser utilizado como parte de nuestra

solución. Dicho patrón de comportamiento corresponde al patrón observador.
El patrón observador es útil cuando se tienen relaciones de dependencias uno-a-muchos, los cuales requieren que un objeto notifique a otros sobre el cambio en su estado, permitiendo a dichos objetos registrar dinámicamente dependencia entre ellos. De esta forma, un objeto puede modificar a otros objetos a través de los cambios de estado que estos sufren [16].

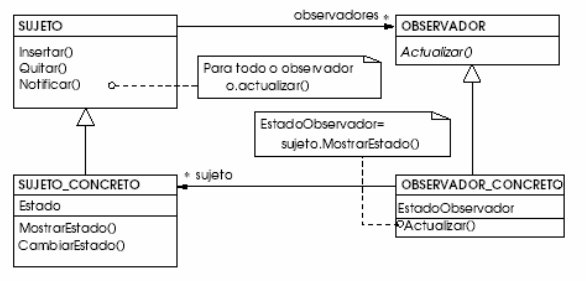

Fig 3. Clase Sujeto_Concreto y Observador

La Figura 3 muestra el diagrama general de clases del patrón Observador [16]. La clase Sujeto_Concreto es quien recibe las notificaciones. En la clase Observador_Concreto es donde ocurren los cambios de estado que son notificados a las clases observadoras.

4.2.2 El patrón observador orientado a aspectos

En nuestra arquitectura, el sujeto concreto está definido por el aspecto NotifyofChange, el cual es una instancia de la clase observadora, la cual queda definida por el aspecto abstracto AbstractNotifyOfChange. El método actualizar estado corresponde al método UpDateObserver de nuestra clase concreta (NotifyofChange). El método UpDateObserver se encarga de notificar si los objetos metadata e historia han cambiado. Como podemos ver en la figura 4, nuestra clase abstracta nos proporciona dos pointcut, con el objeto de manejar más eficientemente la concurrencia a través de los estados de los monitores (los objetos metadata e historia).

```
01 public abstract aspect AbstractNotifyOfChange{
02 protected interface Subject{ }
03 protected interface Observer{ }

04 private WeakHashMap perSubjectObservers;

05 protected List getObservers(Subject s){
06  if(perSubjectObservers==null)
07    perSubjectObservers=new WeakHashMap();

08  List observers=(List)perSubjectObservers.get(s);
09   if(observers==null){
10    observers=new LinkedList();
11    perSubjectObservers.put(s,observers);
12   }

13  return observers;
14 }//fin de getObservers

15 public void addObserver(Subject s,Observer o){
16   getObservers(s).add(o);
17 }//fin de removeObserver

18 public void removeObserver(Subject s,Observer o){
19   getObservers(s).remove(o);
20 }//fin de removeObserver

21 abstract protected void updateObserver(Subject s,
Observer o);
22 abstract protected pointcut changes(Subject s);

23 before(Subject s): changes(s){

24 }//fin del advice before

25 after(Subject s): changes(s){
26  Iterator iter=getObservers(s).iterator();
27  while(iter.hasNext())
28    updateObserver(s,(Observer)iter.next());

29 }//fin del advice after

30}// fin del aspecto abstracto
/* * Fin  A abstracto  **/

/*** aspecto NofifyOfChange **/

31 public aspect NotifyofChange extends
AbstractNotifyOfChange {

32  static int busy=0;
33  static int exist=0;
34  Screen a;

35  declare parents: MetaData implements Subject;
36  declare parents: Historia implements Subject;
37  declare parents: Screen implements Observer;


38  protected pointcut changes(Subject s): target(s) &&
```

```
(call(void metadata.Guardar(String )) || call(void
historia.Guardar(string)));

39  before(Subject p):changes(p){
40    if(exist==0  && busy==0){
41    busy=1;
42    exist=1;
43    addObserver(p,a);}
44    else if(busy==1){
45          while(busy==1);
46        }
47        else if(busy==0){
48              busy=1;
49             }
50   }

51   after(Subject p):changes(p){
52    updateObserver(p,a);
53     }

54   protected void updateObserver(Subject p,Observer s){

55     busy=0;
56     ((Screen)s).display("Se ha producido un cambio en los
objetos");
57  }

  }// Fin de la clase extendida AbstractNotifyOfChange
```

Fig 4. Aspecto de Notificación de Cambios

El before advice (línea 39) se encarga de verificar el estado de los objetos (metadata e historia), si su estado es "no accesado"(línea 40, cuando la variable busy está en 0), entonces éste cambia el estado a "accesado" (línea 41, busy vale 1). Si el estado es "accesado", entonces éste espera a que se produzca el cambio de estado (a través de la línea 45), para poder acceder a dichos objetos. Por otro lado, el after advice (línea 51) se encarga de liberar el objeto, cambiando el estado a "no accesado" (línea 55) una vez que éste a terminado de ejecutar los métodos de las clases correspondientes. El objetivo de la variable exist (línea 33), es el poder agregar sólo una vez al observador, lo cual sucede cuando se alcanza por pimera vez el before advine (línea 39). Por otro lado, podemos observar que se ha instancia un objeto de tipo Screen (línea 34). Dicho objeto de tipo Screen tiene como objeto imprimir en pantalla cuando se ha producido un cambio en los objetos metada e historia. Más aún, dicho objeto de tipo screen juega el rol de observador en esta solución.

5. Discusión del modelo propuesto

Con respecto a promover percepción de usuarios en la Web, hemos agregado algunos nuevos componentes a la arquitectura convencional de la Web, con el objeto de proveer percepción a quienes han visitado una misma página. Nuestro modelo arquitectónico promueve dos grandes beneficios cualitativos sobre la interacción social que se lleva a cabo en la Web. Dichos beneficios son los siguientes; primero, un mínimo de interacción social en la Web debería de proporcionar información con el objeto de encontrar fuentes Web acerca de un tópico o tema en particular. Segundo, si existe una mayor interacción o comunicación entre usuarios que comparten los mismos intereses o inquietudes, debería de existir también una mayor colaboración entre ellos.

Desde el punto de vista de la implementación, esta aproximación particiona el problema de coordinación de múltiples objetos en dos tipos de instancias: las clases secuenciales y los aspectos de comportamiento. Donde los aspectos de comportamiento llegan a ser las responsables de la coordinación de múltiples objetos en un sistema concurrente. Las clases secuenciales son compuestas de la manera tradicional en orientación a objetos, esto es, a través de las relaciones de uso entre los objetos metadata e historia, los cuales son accedidos concurrentemente. Los aspectos de comportamiento están escritos en un lenguaje totalmente diferente al que se escriben las clases secuenciales. El lenguaje en el cual se implementaron los aspectos de comportamiento corresponde a AspectJ. Por otro lado, los aspectos de comportamiento están basados en la incorporación natural del patrón coordinador sobre el patrón observador.

6. Conclusiones

En este artículo, se ha presentado una arquitectura eficiente que promueve percepción de usuario en la Web. Dicha eficiencia reside en la utilización de la programación orientada a aspectos.

Este modelo nace como una respuesta a la necesidad de un manejo eficiente y novedoso ante el problema de sistemas concurrentes orientados a objetos. Por otro lado, también nace como otra opción de los sistemas actuales que proveen mecanismos de percepción en la Web, sobre todo cuando las personas ya han visitado un sitio Web. Otro punto interesante de esta arquitectura es

que esta no modifica el modelo de funcionamiento actual de la Web. Sin embargo, extiende su funcionamiento.

Es importante destacar que nuestra aplicación corresponde a una aplicación pequeña donde se ha incorporado la programación orientada a aspectos. Sin embargo, se ha mostrado que resulta útil e intuitivo utilizar este nuevo paradigma de programación, ya que éste permite generar código reutilizable y extendible de manera natural sin cambiar el código original.

La separación de restricciones del código secuencial utilizando el paradigma orientado a aspectos parece ser una aproximación prominente. Ésta alcanza los objetivos de reducir drásticamente el código complejo entre las clases secuenciales y los aspectos de comportamiento en ambientes concurrentes. En resumen, separando las restricciones de coordinación de múltiples objetos de las clases secuenciales, se ha alcanzado una instancia, donde la reutilización y extensión del comportamiento de múltiples objetos utilizando aspectos es factible

Referencias

[1] L. Sproull, S. Kiesler, S. : Connections: New ways of working in the networked organization. *Cambridge, MA*: MIT Press. (1991)

[2] E. Shapiro: Virtual Places - A Foundation for Human Interaction. *In Proceedings of WWW2, the 2nd International World Wide Web Conference*, Chicago, IL. May, (1994).

[3] C. Gutiérrez-Soto, L. Guerrero, C. Collazos: An Architectural Model to Promote User Awareness on the Web, *ICWE'04*, Munich, Germany, pp. July, 605-606 (2004)

[4] A. Soro, I. Marcialis, D. Carboni, G. Paddeu: WebRogue: rendezvous in a web place. *International Journal of Web Based Communities* vol 3(4) pp. 448-459 (2007)

[5] P. Dourish, V. Bellotti: Awareness and Coordination in Shared Workpaces. *Proc. ACM Conference on Computer Supported Cooperative Work (CSCW)*, Toronto, ACM Press (1992)

[6] S. Sarin, I. Greif: Computer-Based Real-Time Conferencing System,. *IEEE Computer 18*, pp. 33-45 Oct. (1985)

[7] M. Beaudouin-Lafon, A. Karsenty: Transparency and Awareness in Real-Time Groupware System. *Proceedings of the ACM Symposium on User Interface Software and Technology* UIST'92, Monterrey, CA, ACM Press, New York, pp.171-180 (1992)

[8] A. Girgensohn, A. Lee, K. Scuehter: Experiences in developing collaborative applications using the World Wide Web "shell". Proc. Of Hypertext'96, *ACM Press*, (1996)

[9] K. Tollmar, O. Sandor, A. Schoemer: A. Supporting Social Awarenees @ Work Desing and Experience. *Proceeding of CSCW'96*, USA, pp. 298-307 (1996)

[10] S. Greneberg, M. Roseman: GroupWeb: A WWW Browser as Real Time Groupware, in *ACM SICGHI'96* Conference on Human Factors in Computing System, Companion Proceedings, pp. 271-272, (1996)

[11] S. Benford, C. Brown, G. Reynard, C. Greenhalgh: Shared Spaces: Transportation, Artificiality and Spatiality, Proc. CSCW'96 Boston, Massachusetts, *ACM Press*, 16-20 November, pp 77-85 (1996).

[12] A. Girgensohn, A. Lee, K. Scuehter: Experiences in developing collaborative applications using the World Wide Web "shell". Proc. Of Hypertext'96, *ACM Press*, (1996)

[13] H. Ishii, B. Ulmer: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. Proc. Of CHI´97, *ACM Press*, (1997)

[14] J. Morrison, P. Pirolli, S. Card: A Taxonomic Analysis of What World Wide Web Activities Significantly Impact People's Decisions and Actions. Proc. CHI 2001, Extend Abstract, pp. 163-164 (2001)

[15] A. Sellen, R. Murphy, K. Shaw: How Knowledge Workes Use the Web. Conference Proceedings CHI'02, Minneapolis, Minnesota, USA, *ACM SIGCHI* (2002)

[16] E. Gamma, R. Helm, R. Johnson,. et al : *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison-Wesley, (1994)

[17] C. Jaques: Concurrent Object-Oriented Programming. *In Communications of the ACM*. Vol. 36. No. 9. September (1993)

[18] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, J. Irwin: Aspect-Oriented Programming. *XEROX PARC Technical Report*, SPL97-008 P9710042, February (1997)