Modelado Conceptual de aplicaciones adaptivas y proactivas en OO-H

Irene Garrigós[1], Cristina Cachero[1], and Jaime Gómez[1]

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante. SPAIN
{igarrigos,ccachero,jgomez}@dlsi.ua.es

Abstract La personalización de entornos ha suscitado un gran interés desde el punto de vista de los métodos de modelado conceptual de aplicaciones web. Ello es debido al efecto que tienen las políticas de personalización sobre todo el ciclo de desarrollo de la aplicación, desde la recogida de requisitos a la propia arquitectura de ejecución. La inclusión de estrategias de personalización dinámica es una cuestión no resuelta a nivel de modelado conceptual. Desde OO-H (*Object Oriented Hypermedia*) creemos que el tratamiento de las propiedades relevantes que guian la construcción de un modelo conceptual con soporte de personalización es el camino más adecuado para diseñar este tipo de característica en aplicaciones web. Actualmente, la mayoría de los esfuerzos en este sentido se plantean a nivel de implementación proporcionando soluciones ad-hoc. Este artículo propone ciertas extensiones a nivel de modelado conceptual que, embebidas en los modelos de navegación y presentación de OO-H, permiten capturar una especificación XML en base a la cual se definen las reglas de personalización dinámica de una aplicación web. A partir de esta especificación, una arquitectura de ejecución dinámica basada en un motor de reglas es capaz de interpretar la parte variable de la aplicación.

1 Introducción

La mayoría de los métodos, técnicas y procesos ingenieriles que pertenecen a la Ingeniería Web [7] intentan hacer más sencilla la comprensión, desarrollo, evolución y mantenimiento de una aplicación web. Esto ha supuesto en muchos casos la extensión de técnicas aplicadas en la ingeniería del software 'tradicional' con nuevos constructores y vistas hipermediales [10, 14, 4, 15] que abordan el problema de la navegación/presentación del usuario a través del espacio de información.

En este contexto, una de las características que más interés está suscitando en la comunidad científica es como tratar adecuadamente nuevas necesidades web, como la identificación de perfiles que modelan los distintos tipos de usuarios, incluyendo estrategias tanto estáticas como dinámicas. La mayoría de los trabajos en este campo han estado centrados en proporcionar soluciones de implementación ad-hoc para dominios muy concretos. A nivel de modelado conceptual sólo las características de personalización estática han sido tratadas por algunos métodos [4, 14, 10] mediante la incorporación de perfiles de usuario en diagramas de navegación.

Este artículo presenta la respuesta que ofrece el Método Hipermedial Orientado al Objeto (*Object Oriented Hypermedia (OO-H)* [8]) para modelar las aplicaciones web con soporte de personalización dinámica. Concretamente personalización dinámica adaptiva y proactiva. Las aplicaciones adaptivas (*Adaptive*)[9] utilizan información recogida y analizada por la propia aplicación de manera transparente para el usuario. Ejemplos típicos de este tipo de aplicaciones son los sistemas de recomendación (e.g. [1]) utilizados como apoyo en numerosas aplicaciones de comercio electrónico, que utilizan mecanismos de filtrado colaborativo y reglas de asociación. Las aplicaciones proactivas (*Proactive*) [5] son capaces de detectar *de motu propio* cambios en el entorno y reaccionar en consecuencia, actuando de este modo como Observadoras [6] del entorno. La principal diferencia respecto de las aplicaciones adaptivas es que, mientras que en éstas cualquier cambio es causado por una acción del usuario, una aplicación proactiva puede cambiar la vista del usuario sin necesidad de que éste realice ninguna acción. Implica por tanto tecnología *push*, frente al tradicional esquema *pull* de las

aplicaciones web. Un ejemplo de comportamiento proactivo es la capacidad por parte de la aplicación de detectar un cambio en las condiciones de interacción (por ejemplo en la localización de un dispositivo de acceso móvil), y la modificación automática de la vista de información ofertada en función de dichas condiciones.

Este artículo propone ciertas extensiones a nivel de modelado conceptual que, embebidas en los modelos de navegación y presentación de OO-H, permiten capturar una especificación XML en base a la cual se definen las reglas de personalización dinámica de una aplicación web. A partir de esta especificación, una arquitectura de ejecución dinámica basada en un motor de reglas es capaz de interpretar la parte variable de la aplicación. Más concretamente la capa de lógica de aplicación se segmenta en dos subcapas: una capa que especifica la parte no variable de la aplicación y otra capa que especifica la parte variable. Dicha parte variable es sensible a la actividad del usuario en el sistema que se registra en un repositorio.

El artículo está estructurado como sigue: en la sección 2, para contextualizar este trabajo, se presentan las diferentes técnicas de personalización dinámica. En concreto las reglas de asociación, técnica elegida por OO-H. A continuación la sección 3 presenta la arquitectura de la personalización y su efecto en los modelos de navegación/presentación. Además, en esta sección se detalla el metamodelo de personalización que permite estructurar la información del usuario para especificar las reglas de asociación en base a un esquema XML [16]. También se presenta la estructura y semántica de esta plantilla utilizando un ejemplo comprensivo. Por último, en la sección 4 se presenta las conclusiones y trabajos futuros.

2 Técnicas de personalización dinámica

En el siguiente cuadro (ver Figura 1) podemos ver una clasificación de las técnicas de personalización dinámica [3]:

Filtrado simple	***Filtrado por contenido***	***Filtrado Colaborativo***	***Perfilado dinámico***	***Reglas de asociación***
Definición de grupos de usuarios y asociación de vistas específicas de la aplicación a cada grupo.	*Definición de categorías de interés.* *Pertenencia o no de los objetos a dichas categorías.* *Se muestran los objetos de los grupos de categorías que tengan interés para el usuario.*	*Los usuarios son asignados (mediante algún algoritmo analiza sus preferencias) a grupos basados en similaridades de acceso al contenido de páginas web.*	*Permite la personalización de la aplicación en base a la actividad del usuario en otras aplicaciones web* *La información es obtenida mediante un identificador global suministrado por el propio usuario de manera implícita*	*Definición en forma de ficheros externos de criterios q cuando se cumplen por un usuario determinado y opcionalmente se produce un evento, desencadenan una acción determinada.* ***Reglas Reactivas:*** *evento causado directamente por el usuario.* ***Reglas Proactivas:*** *Cambio en el contexto.*
Ejemplo: *Según la ubicación del usuario (a partir de la URL) le mostramos la aplicación en uno u otro idioma al navegante.*	*Requiere que no intervenga mucho la subjetividad del usuario.* ***Ejemplo:*** *Clasificación de documentos por categorías.*	*Interviene en gran medida la subjetividad del usuario.* ***Ejemplo****:* *Usado en sistemas de recomendación como www.Amazon.com*	*Limita los problemas asociados a la seguridad y privacidad de los datos*	***Ejemplo:*** *Se pueden definir perfiles de usuario con este tipo de reglas (ej. patrones de navegación de usuario)*

Figure1. *Tipos de técnicas de personalización dinámica*

Las reglas de asociación pueden ser válidas para aplicar esquemas de personalización a grupos de usuarios (filtrado simple). El filtrado basado en contenidos es adecuado cuando no se quiere depender de una evaluación del usuario, pero requiere la capacidad de analizar los objetos, lo cual hoy en día es difícil con determinado tipo de información (imágenes, vídeos) y está limitado en cuanto a semántica. El filtrado colaborativo complementa al filtrado basado en contenidos: obtiene información del grupo en vez del simple emparejamiento de patrones. Esta técnica conlleva un alto coste computacional y un bajo grado de fiabilidad ante volúmen de informacion reducido ya que requiere una gran cantidad de usuarios para conseguir información válida. La personalización basada en reglas usa información específica de los individuos que visitan un sitio web para mostrar el contenido de forma más precisa. La información que describe las necesidades, intereses, preferencias y motivaciones de los visitantes se captura en forma de perfiles, patrones de uso histórico y vectores de clicks. Las reglas de asociación se usan para personalizar la experiencia del usuario; estas reglas se aplican luego para proveer información útil y recomendaciones relevantes para los visitantes.

Usando personalización basada en reglas un sitio web puede informar a sus visitantes de promociones interesantes, ofrecerles descuentos basados en su historial de compras, etc. Los mecanismos de reglas se usan para personalización en muchos tipos diferentes de aplicaciones web porque permiten a los sitios web interactuar inteligentemente con los clientes. Dando a los usuarios y administradores la capacidad de definir reglas de asociación para personalizar la experiencia de usuario, puede adoptarse una aproximación muy personal de marketing y venta, donde las necesidades, el comportamiento y los intereses del usuario determinan la información que se muestra, las oportunidades que se presentan y las recomendaciones que se hacen.

La principal ventaja de las reglas de asociación es la disminución del acoplamiento (con la consiguiente disminución del coste de mantenimiento y evolución) de las actividades de personalización respecto al resto de actividades del método. Por todos estos motivos OO-H se decide por utilizar las reglas de asociación.

El uso de este tipo de reglas requiere incorporar en la arquitectura de la aplicación de un motor que soporte la evaluación de las mismas. A continuación, se presentan los cambios respecto a la arquitectura inicial de aplicaciones basadas en OO-H. La estructura y comportamiento concreto de las reglas en el marco de OO-H será visto en la sección 3.3.

3 Modelado de la Personalización Dinámica en OO-H

3.1 Arquitectura de la Personalización

La arquitectura de OO-H presentada en [8] necesita ser modificada sensiblemente para soportar la personalización dinámica. En la figura 2 se presenta la arquitectura modificada. Se puede observar como las características de personalización se capturan en el diagrama de navegación/presentación mediante un conjunto de reglas de asociación. A la arquitectura se le añade un motor que permite interpretar esas reglas en tiempo de ejecución. De esta forma se consigue que el diseño y generación de la lógica de navegación se especifique en dos partes. Una parte estable que es independiente de las propiedades de personalización y una parte variable que soporta el tratamiento de tales reglas.

Al modelar las partes variable y estable de la aplicación de forma separada ya desde los primeros estados del ciclo de vida del software, la naturaleza dinámica de la ubicuidad de las aplicaciones web puede tratarse mejor, como también la reusabilidad y la realización de cambios. La justificación de este planteamiento es similar a las razones argumentadas para separar las políticas de negocio de aplicaciones orientadas a objeto. [13]. El soporte de esta arquitectura se consigue a partir de un metamodelo que permite capturar las propiedades relevantes de personalización y que presenta a continuación.

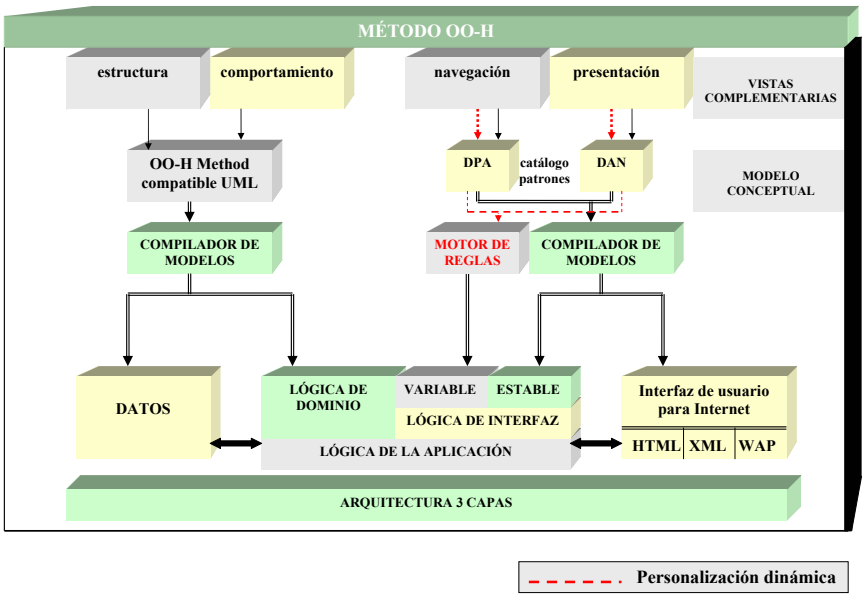

Figure2. *Arquitectura OO-H con personalización dinámica*

3.2 El Metamodelo de Personalización en OO-H

La estrategia de modelado de la personalización debe definirse en función de una serie de estructuras de información. Esta información se almacena en OO-H en un repositorio que se presentó en [3] y que ha variado ligeramente en algunos atributos (ver Fig. 3). Este respositorio contiene el conjunto inicial de elementos básicos de información sobre los que se puede establecer la política de personalización deseada. El diseñador puede incluir y conectar este marco de trabajo con cualquier modelo de interfaz OO-H al que quiera dotar de capacidad de personalización. A partir de ahí, OO-H permite la extensión de dicho repositorio con las características particulares que requiera la aplicación, por lo tanto, no es un marco cerrado.

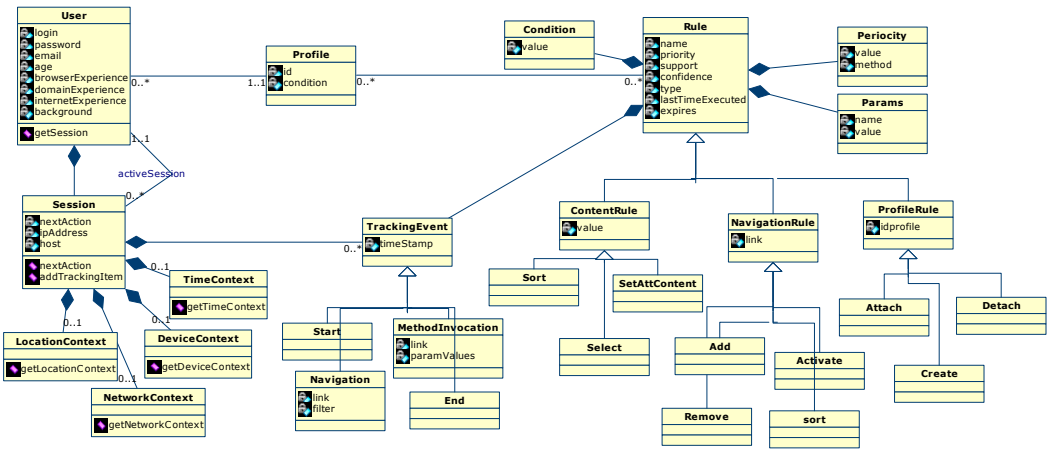

Figure3. *Marco de trabajo de Personalización*

El metamodelo (ver Fig 3), estructura el modelado de la personalización en OO-H en tres partes que son perfiles de usuario, información de contexto y reglas de asociación.

El **Perfil de Usuario** está determinado tanto por los datos proporcionados explícitamente por el usuario (e.g. formulario de entrada) como por la información recogida de forma implícita a partir de su actividad en el sistema (transparente al usuario). Este perfil puede capturar aspectos muy diversos como preferencias, características, problemas técnicos, patrones de comportamiento (búsquedas realizadas, páginas visitadas, productos o servicios adquiridos...), etc.

En la Fig 3 podemos observar cómo el perfil de usuario se almacena en la clase *User*, que contiene un conjunto de objetos *Session*, uno por cada sesión que haya abierto el usuario en el sistema. El usuario puede estar preasignado a un perfil (clase *Profile*) o ser asignado dinámicamente en cualquier momento durante su sesión.

Con el fin de proporcionar información sobre la actividad del usuario en el sistema a los algoritmos de filtrado, OO-H define cuatro tipos de evento de usuario, correspondientes a las acciones de usuario que pueden ser registradas actualmente de manera automática en OO-H. Este registro se produce mediante la instanciación de distintos tipos de objetos de seguimiento (*TrackingEvents*, ver Fig. 3), a saber:

- *Start*: implica la entrada de un nuevo usuario en el sistema.
- *Navigation*: implica la activación de un enlace de navegación en la interfaz asociada. Una acción de navegación guarda su contexto, es decir, las condiciones de filtro que tenía asociadas cuando se activó.
- *Method Invocation*: implica la invocación de un servicio ofertado por la lógica. Este tipo de acción supone guardar tanto el enlace de servicio activado como los valores introducidos en los parámetros para esa invocación.
- *End*: implica la salida del usuario del sistema.

La inclusión de estos objetos en la sesión del usuario se realiza mediante la asociación de eventos de creación de objetos de seguimiento a los distintos tipos de enlace que ofrece OO-H. Vamos a ver cómo: un enlace de entrada a la aplicación, etiquetado como *Entry Point*, llevará asociado un evento de creación de un objeto de seguimiento de tipo *Start*, que se lanza cada vez que un nuevo usuario activa el enlace de entrada a la aplicación.

Del mismo modo, con la activación de un enlace de servicio se crea un objeto de seguimiento de tipo *MethodInvocation*, que almacena en su atributo *link* el id del enlace activado.

Siguiendo con la misma filosofía, la acción de activar cualquier enlace de navegación en OO-H provoca, paralelamente al efecto de navegación, la creación de un objeto de seguimiento de tipo *Navigation*, que almacena, además del enlace en cuestión, la información de contexto (objeto concreto desde el cual se navegó, restricciones en la población destino del enlace, etc) de dicha navegación.

Por último, la instanciación de un objeto de seguimiento de tipo *End* (que, como efecto colateral, finaliza la sesión del usuario en el sistema) se puede producir de dos modos: bien por la activación de un enlace definido como 'de salida' en OO-H (*Exit Point*) o bien de manera automática, cuando la aplicación detecta un período de inactividad en el sistema superior a un valor umbral (en este momento establecido en 5 minutos).

Los objetos de seguimiento son especialmente útiles para la detección de nuevos patrones de navegación de usuario, como puede verse en [11]. Además, y debido a que en OO-H los objetos de seguimiento no guardan URL's sino enlaces conceptuales (ver Fig. 3), las facilidades que ofrecen los métodos de modelado hipermedial de una categorización exhaustiva de estos enlaces (por ejemplo identificación de determinados enlaces de servicio como enlaces de tipo 'compra', o de determinados enlaces de navegación como enlaces de tipo 'descarga' [2]) permite definir estrategias mucho más

precisas.

El metamodelo ofrece, además, estructuras para el almacenamiento del **contexto** en el que se produce la interacción usuario-aplicación. Este contexto se incluye como parte de la sesión del usuario, y en OO-H se divide en: (ver Fig. 3):

- *Network Context*: Comprende información sobre la red (latencia, ancho de banda etc de la conexión).
- *Time Context*: Permite adaptar la aplicación con respecto a ciertas restricciones de horario como horas de apertura de tiendas y horarios de transporte público (fecha y hora local de la conexión).
- *Device Context*: Incluye información de los dispositivos relevantes (software instalado en ellos) y las clases de dispositivo (palm, portátil, PC...).
- *Location Context*: Captura información sobre la localización de acceder a una aplicación (ubicación del cliente).

Obviamente, cada aplicación usará el subconjunto de información que considere relevante (o al que tenga acceso). Este tipo de información es muy utilizado para el modelado de reglas proactivas.

Por último, el repositorio también proporciona una estructura de almacenamiento de las **Reglas de asociación** que a continuación se presentan con todo detalle en la siguiente subsección. Como mostraremos más adelante, este tipo de reglas permiten capturar las propiedades de personalización embebidas directamente en los modelos de OO-H.

3.3 Especificación de las reglas de activación

OO-H incorpora un mecanismo externo para modelar la personalización de las aplicaciones adaptivas y proactivas en forma de reglas de activación. Una ventaja de usar estas reglas es que al estar definidas en ficheros externos pueden modificarse al instante de forma independiente del resto de la aplicación. Como vemos en la Fig. 3, estas reglas se categorizan, en función del resultado que su activación tiene en el sistema, en reglas de contenido (*ContentRule*), de navegación (*NavigationRule*) y de definición de perfiles (*ProfileRule*).

OO-H da soporte a la especificación de las reglas, mediante el esquema XML [16] que se presenta a continuación. Es importante hacer notar que la arquitectura de ejecución de las aplicaciones generadas a partir de modelos OO-H permite que este esquema pueda ser modificado y reprocesado por la aplicación sin necesidad de recompilar el resto de módulos.

```
<Schema name="PersonalizationSchema" xmlns="urn:schemas-microsoft-com:xml-data"
 xmlns:dt="urn:schema-microsoft-com:datatypes">
<ElementType name="profile" content="mixed" order="seq">
    <element type="rule" minOccurs="1" maxOccurs="*"/>
    <attribute type="profileId"/>
    <attribute type="profileCondition"/>
</ElementType>
<ElementType name="rule" content="mixed" order="seq">
    <attribute type="name"/>
    <attribute type="priority"/>
    <attribute type="support"/>
    <attribute type="confidence"/>
    <attribute type="typeRule"/>
    <attribute type="lastTimeExecuted"/>
    <attribute type="expires"/>
    <element type="params" minOccurs="1" maxOccurs="1"/>
    <element type="periocity" minOccurs="0" maxOccurs="1"/>
    <element type="event" minOccurs="0" maxOccurs="1"/>
    <element type="condition" minOccurs="1" maxOccurs="1"/>
    <element type="action" minOccurs="1" maxOccurs="*"/>
</ElementType>

<ElementType name="params" content="eltOnly" order="seq">
    <element type="param" minOccurs="1" maxOccurs="*"/>
</ElementType>
<ElementType name="param" content="textOnly" order="seq">
    <attribute type="name"/>
```

```
        <attribute type="value"/>
    </ElementType>
    <ElementType name="event" content="textOnly" order="seq">
        <attribute type="timeStamp"/>
        <attribute type="mode"/>
        <element type="EventStart"/>
        <element type="EventNavigation"/>
        <element type="EventMInvocation"/>
        <element type="EventEnd"/>
    </ElementType>
    <ElementType name="EventStart">
        <attribute type="Start"/>
    </ElementType>
    <ElementType name="EventNavigation">
        <attribute type="Navigation"/>
        <attribute type="link"/>
        <attribute type="filter"/>
    </ElementType>
    <ElementType name="EventMInvocation">
        <attribute type="MInvocation"/>
        <attribute type="link"/>
        <attribute type="paramValues"/>
    </ElementType>
    <ElementType name="EventEnd">
        <attribute type="End"/>
    </ElementType>
    <ElementType name="periocity" content="textOnly" order="seq">
        <attribute type="periocityValue"/>
        <attribute type="method"/>
    </ElementType>
    <ElementType name="condition" content="textOnly" order="seq">
        <attribute type="value"/>
    </ElementType>
    <ElementType name="action" content="eltOnly" order="seq">
        <element type="contentRule" minOccurs="0" maxOccurs="1"/>
        <element type="navigationRule" minOccurs="0" maxOccurs="1"/>
        <element type="profileRule" minOccurs="0" maxOccurs="1"/>
    </ElementType>
    <ElementType name="contentRule" content="textOnly" order="seq">
        <attribute type="actionContentR"/>
        <attribute type="value"/>
    </ElementType>
    <ElementType name="navigationRule" content="textOnly" order="seq">
        <attribute type="actionNavigationR"/>
        <attribute type="value"/>
    </ElementType>
    <ElementType name="profileRule" content="textOnly" order="seq">
        <attribute type="actionProfileR"/>
        <attribute type="profileId"/>
    </ElementType>
    <AttributeType name="profileId" dt:type="char" default="ooh:all"/>
    <AttributeType name="profileCondition" dt:type="char" default="ooh:all"/>
    <AttributeType name="name" dt:type="char"/>
    <AttributeType name="typeRule" dt:type="enumeration" dt:values="content navigation profile"/>
    <AttributeType name="support" dt:type="int"/>
    <AttributeType name="confidence" dt:type="int"/>
    <AttributeType name="priority" dt:type="int"/>
    <AttributeType name="lasTimeExecuted" dt:type="date"/>
    <AttributeType name="expires" dt:type="date"/>
    <AttributeType name="timeStamp" dt:type="enumeration" dt:values="Adquisition Personalization"/>
    <AttributeType name="mode" dt:type="date"/>
    <AttributeType name="value" dt:type="char"/>
    <AttributeType name="link" dt:type="char"/>
    <AttributeType name="filter" dt:type="char"/>
    <AttributeType name="paramValues" dt:type="char"/>
    <AttributeType name="actionContentR" dt:type="enumeration" dt:values="Select SetAttContent Sort"/>
    <AttributeType name="actionNavigationR" dt:type="enumeration" dt:values="Add Remove Activate Sort"/>
    <AttributeType name="actionProfileR" dt:type="enumeration" dt:values="Attach Detach Create"/>
    <AttributeType name="Start" dt:type="enumeration" dt:values="Start"/>
    <AttributeType name="Navigation" dt:type="enumeration" dt:values="Navigation"/>
    <AttributeType name="MethodInvocation" dt:type="enumeration" dt:values="MethodInvocation"/>
    <AttributeType name="End" dt:type="enumeration" dt:values="End"/>
    <AttributeType name="periocityValue" dt:type="char" default="ooh:always"/>
    <AttributeType name="method" dt:type="char"/>
</Schema>
```

En él podemos ver cómo los distintos elementos XML corresponden a clases/atributos del marco presentado en la Fig. 3. En primer lugar, observamos cómo una regla se define en el contexto de un perfil. Para indicar que la regla se debe evaluar sin importar el usuario que entre en el sistema, a

dicha etiqueta se le asocia el valor *"ooh:all"*, predefinido en el contexto de OO-H. Las reglas tienen además varios atributos asociados, y que son [12]:

- Nombre (*name*): nombre de la regla
- Tipo de regla (*type*): tipo de efecto que la activación de la regla causaría en el sistema. Si se trata de una regla que ordena o muestra una selección de atributos es de tipo *Content*. Si es una regla que afecta al modo de navegación es de tipo *Navigation*. Si es una regla de creación de perfiles es de tipo *Profile*.
- Soporte (*support*): porcentaje de veces que se ha activado la regla en nuestra aplicación.
- Confianza (*confidence*): porcentaje de aciertos en la ejecución de la regla. El tipo de política que representa esta regla hace presuponer que el usuario siempre va a estar de acuerdo con ella.
- Prioridad (*priority*): importancia de la regla en el sistema en caso de conflicto o condiciones de carrera.

Cada regla puede tener asociada una sección de parámetros.

El cuerpo de la regla lo constituyen tres secciones (ver Fig. 3): En primer lugar el Evento *Event* determina qué acción de usuario actúa como disparador de la regla. Los posibles eventos que se pueden producir son: que el usuario entre en nuestra aplicación *Start*, que el usuario navegue por la aplicación *Navigation*, que el usuario invoque a un determinado método *MethodInvocation* y por último, que el usuario salga de la aplicación *End*. Alternativamente, se puede establecer una Periocidad de la regla (*Periocity*, que especifica cada cuánto tiempo, de manera automática, la aplicación debe chequear dicha regla. En el contexto de la etiqueta de periocidad, OO-H define un valor especial, *"ooh:always"*, que indica que el cumplimiento de las condiciones de activación se debe comprobar de manera continua. La segunda parte del cuerpo de la regla lo constituye la condición, que especifica una expresión de guardia que permite o inhibe la ejecución de la regla. Por último, la tercera parte de la regla es la acción. Esta etiqueta podrá tomar los siguientes valores según el tipo de regla implementada:

- Regla de tipo *Content*: la acción asociada podrá ser: *Select, SetAttContent, Sort*. En función de si lo que se quiere es seleccionar, modificar u ordenar la información de contenido de la aplicación.
- Regla de tipo *Navigation*: la acción asociada podrá ser: *Add, Remove, Activate o Sort*. En función de si lo que se quiere es añadir, eliminar, activar u ordenar enlaces en la navegación del usuario.
- Regla de tipo *Profile* la acción asociada será: *Attach, Detach o Create*. En función de si lo que se quiere es asociar o desasociar comportamiento específico a perfiles de usuario, o bien crear un nuevo perfil de usuario.

A continuación veremos algunos ejemplos que muestran como modelar la personalización dinámica en un modelo OO-H. Utilizaremos un sistema de información de un video club por internet con clientes, peliculas, ofertas ...

3.4 Ejemplos

En este contexto, un ejemplo de adquisición proactiva sería la detección por parte de la aplicación de la existencia de un plug-in para ver un trozo de la película cuando el usuario quiera información de una película determinada. Si detecta el plug-in, le mostrará un enlace para poder ver el trailer, de lo contrario ocultará esta opción. La regla a utilizar para implementar este caso (**regla 1**) sería de tipo *Navigation* ya que la visualización o no visualización del enlace depende del contexto de dispositivo del usuario, es decir que esta regla afecta al modo de navegación:

```
<TPersonalization>
    ...
    <profile id="ooh:all" condition=''>
        <rule type="navigation" name="ComprobarPlugin" support="12" confidence="100" priority="8"
         expires="1/01/03" lastTimeExecuted="24/05/02">
            <params>
                <param name="pluginvideo" value="session.socio.pluginvideo"/>
            </params>
```

```
            <periocity periocityValue="ooh:always" method="getDeviceContext"/>
            <event mode="Personalization" type="Navigation" link="Ver Peliculas"/>
            <condition value="pluginvideo=1"/>
            <action type="Activate" value="session.pelicula.trailer='visible'"/>
        </rule>
    </profile>
    ...
</TPersonalization>
```

Podemos ver que esta regla se aplica a todos los perfiles. Como parámetro tenemos el atributo *plug-invideo* que es un booleano que indica si el socio tiene o no el plugin necesario. La condición en la que se chequea que el plugin se encuentre en la parte cliente se está comprobando continuamente al tener establecida como periodicidad el valor por defecto *ooh:always*, y de cumplirse la condición, en la acción se especifica la activación de un nuevo enlace para poder visualizar el trailer. Como vemos la forma de adquisición es proactiva, pero la forma de personalización es reactiva, ya que es causada por un evento de tipo *Navigation*, al activar el link "Ver Películas".

Supongamos ahora que al alquilar una película, el sistema nos hace recomendaciones de otras películas con el mismo tema que la que hemos elegido. En este caso lo que tenemos es personalización adaptiva y lo implementamos con una regla de tipo *Content* (**regla 2**) ya que afecta al contenido que se muestra, como explicaremos a continuación:

```
<TPersonalization>
    ...
        <profile id="ooh:all" condition=''>
        <rule type="content" name="Recomendaciones" support="17" confidence="100" priority="12"
            expires="1/01/03" lastTimeExecuted="28/05/02">
            <event mode="Adquisition" type="MethodInvocation" link="Alquilar"/>
            <action mode="Adquisition" type="setAttContent"
            value="session.socio.pelicula.tema=alquilar.pelicula.tema"/>
            <params>
                <param name="temapelicula" value="session.socio.pelicula.tema"/>
            </params>
            <event mode="Personalization" type="Navigation" link="Ver Recomendaciones"/>
            <condition value="pelicula.tema==temapelicula and pelicula.alquilada==0"/>
            <action mode="Personalization" type="select" value="pelicula.tema"/>
        </rule>
    </profile>
    ...
</TPersonalization>
```

Esta regla también se aplica a todos los perfiles. La forma de adquisición es reactiva, el evento que causa la adquisición es de tipo "MethodInvocation". La acción de adquisición que se produce es guardar el tema de la última película que ha alquilado el socio. Tenemos como parámetro el tema de la película. El evento de personalización es de tipo Navigation, lo que implica que el enlace *Ver Recomendaciones* tiene que estar activo. Además se tiene que cumplir la condición: que el tema de la película que ha sido alquilada sea del mismo tema que las peliculas que vamos a recomendar y que esa película no la haya alquilado ya el usuario activo. De cumplirse estas premisas, se ejecutará la acción: se muestra el contenido de las películas con las condiciones impuestas.

Supongamos que tenemos otra regla (**regla 3**) que según al videoclub al que pertenezca un usuario, le incluye en un determinado perfil, ya que cada videoclub tiene un número diferente de películas (según su demanda) y esto se tiene en cuenta a la hora de hacer ofertas. En este caso, tenemos personalización es adaptiva ya que se realiza de manera transparente para el usuario y se produce tras un evento del usuario. Debemos usar dos reglas de tipo *Profile*, una de creación de un determinado perfil, y otra de asignación de los usuarios a ese perfil:

```
<TPersonalization>
    ...
    <profile id="ooh:all" condition=''>
        <rule type="profile" name="PerfilCPostal1" support="22" confidence="100" priority="9"
         expires="1/01/05" lastTimeExecuted="11/09/02">
            <params>
                <param name="codpostal" value="session.socio.codpostal"/>
            </params>
            <event mode="Personalization" type="Navigation" link="Autentification"/>
            <condition value="codpostal=03005"/>
```

```
            <action mode="Personalization" type="Create" idprofile="PC1"/>
        </rule>
    </profile>
    ...
</TPersonalization>


<TPersonalization>
    ...
    <profile id="ooh:all" condition=''>
        <rule type="profile" name="AddPerfilCPostal1" support="22" confidence="100" priority="10"
         expires="31/01/04" lastTimeExecuted="12/10/02">
            <params>
                <param name="codpostal" value="session.socio.codpostal"/>
            </params>
            <event mode="Personalization" type="Navigation" link="Autentification"/>
            <condition value="codpostal=03005 and not PerfilCPostal1->isEmpty"/>
            <action mode="Personalization" type="Attach" idprofile="PC1"/>
        </rule>
    </profile>
    ...
</TPersonalization>
```

Ambas reglas afectan a todos los perfiles y tienen como parámetro el código postal. El evento por el cual se dispara la regla es de tipo *Navigation* y esto indica que para que se dispare esta regla tiene que estar activo el enlace "Autentification". En la primera de las reglas, se crea el perfil especificado (*action: create*). En la segunda regla, si se cumple la condición especificada (que el socio tenga el código postal especificado y que exista el perfil al que lo vamos a asociar) se incluirá el socio en el perfil definido (*action: attach*).

Por último supongamos que queremos dar soporte para ofertas inteligentes. Es decir, según el número de alquileres que realices al mes y que días los realices tendrás una serie de alquileres gratuitos. Hay un seguimiento de los alquileres realizados de manera que te informan de qué ofertas puedes beneficiarte. Este tipo de personalización es adaptiva. Representamos esta opción con reglas de tipo 'content'. Supongamos que el detalle de la regla fuese "Cada 5 alquileres en un mes le damos 1 alquiler gratis con un máximo de 2 por mes " (**regla 4**). La forma de especificar tal situación en el metamodelo sería:

```
<TPersonalization>
    ...
    <profile id="PC1" condition="codpostal=03005">
        <rule type="content" name="Ofertas" support="20" confidence="100" priority="15"
         expires="1/01/05" lastTimeExecuted="7/11/02">
            <event mode="Adquisition" type="MethodInvocation" link="Alquilar"/>
            <action mode="Adquisition" type="setAttContent"
            value="session.socio.numalquileres=session.socio.numalquileres+1"/>
            <params>
                <param name="numrentals" value="session.socio.numalquileres"/>
            </params>
            <event mode="Personalization" type="Navigation" link="Ver Ofertas"/>
            <condition value="numrentals=5 or numrentals=10"/>
            <action mode="Personalization" type="Select" value="oferta.id=03"/>
        </rule>
    </profile>
    ...
</TPersonalization>
```

En este caso la regla afecta sólo a los usuarios de perfil 'PerfilCPostal1'. Además, tenemos un parámetro que es el número de alquileres realizados al mes, el cual se obtiene de manera reactiva, ante la invocación del servicio "Alquilar" mediante una acción de tipo "SetAttContent". La personalización se dispara ante un evento de tipo *Navigation* ante la activación del enlace "Ver Ofertas". Como condición de la regla ponemos que este número igual a 5 o igual a 10, de esta forma controlamos que solo se aplique en esos dos casos. Si se cumple la condición se ejecutará la acción, que sería darle un alquiler gratis, por lo que se añade un link con la oferta de la que se puede beneficiar el socio.

Una vez tenemos especificadas las reglas vamos a explicar el proceso a través del cual las hacemos efectivas en los modelos de OO-H. En este caso, todas las reglas pueden especificarse a nivel de diagrama de navegación ya que ninguna de ellas afecta a aspectos de presentación.

Primeramente se ha de detectar el elemento de modelado que se ve afectado por cada regla. Estos pueden ser: un atributo, una clase o un enlace. Cada elemento NAD dispone de una propiedad para asociar las reglas (rules spec). En el cuerpo de la regla se especifica la condición/acción. Las reglas que llevan asociado un evento que las dispara siempre se asocian a enlaces de navegación (enlaces de entrada, internos, de servicio, de respuesta...)

En la figura 4 podemos ver el diagrama de navegación en el que se basa este sistema.

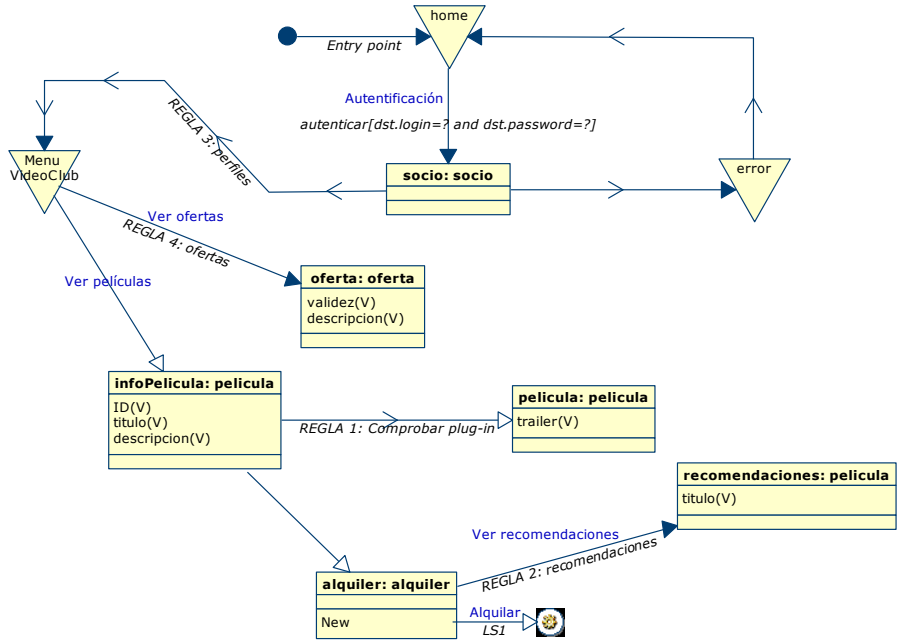

Figure4. *Diagrama de navegación del videoclub*

En el NAD adjunto podemos ver la correspondencia de las distintas reglas especificadas con elementos de modelado. Como dijimos anteriormente, estas reglas estas asociadas a enlaces.

La forma de representar estas reglas en el NAD se basa en sentencias OCL extendidas que capturan la semántica especificada en los esquemas XML presentados anteriormente. La formalización de estas sentencias puede encontrarse en [8].

El NAD tiene como punto de entrada un enlace de requisito que primeramente presenta al usuario un formulario para que introduzca sus parámetros de conexión al sistema *(login y password)*. Si es un usuario autorizado se le presenta un índice con los títulos de las películas disponibles según su perfil (Regla 3). Adicionalmente, se le presentarán al usuario mediante un enlace las ofertas de las que puede beneficiarse (Regla 4). Cuando el usuario seleccione una película, tendrá acceso a la información detallada de la misma con la posibilidad de si es el caso (regla 1) ver el trailer de película. Finalmente, el usuario podrá alquilar la película y tras realizar esta acción se le presentará mediante otro enlace recomendaciones (regla 2) de otras películas similares a la alquilada.

En un proceso posterior el NAD puede ser compilado para obtener la especificación XML que es interpretada por el motor de reglas para dotar de soporte de personalización a la aplicación diseñada.

4 Conclusiones y trabajos futuros

En este artículo se ha presentado el modelado de las aplicaciones adaptivas y proactivas en OO-H. Para el soporte de la personalización en tiempo de ejecución, la arquitectura OO-H se apoya en un motor de reglas de asociación, que se usa para modelar la parte variable de la lógica cliente de la aplicación. Las reglas se aplican a las vistas de presentación y de navegación, y se ven reflejadas en sus correspondientes modelos conceptuales (DPA, DAN). El principal motivo de esta elección es que las reglas de asociación mantienen la funcionalidad de las demás técnicas existentes y la disminución del acoplamiento de las actividades de personalización con el resto de las actividades del método. OO-H define estas reglas, siguiendo su filosofía, mediante un esquema XML. La arquitectura de ejecución de las aplicaciones generadas a partir de modelos OO-H permite que este esquema pueda ser modificado y reprocesado por la aplicación sin necesidad de recompilar el resto de módulos. Además OO-H cuenta con un metamodelo que estructura el conjunto inicial de elementos de información sobre los que se puede establecer la política de personalización deseada.

La validez de esta estrategia ha sido probada mediante la aplicación de ejemplos de distintas reglas de personalización para un sistema de información de un video club por internet.

Como trabajo futuro queda por determinar la forma de especificar algunas de estas reglas en el diagrama de presentación, así como estudiar la forma de incluir en OO-H estándares de privacidad e intercambio de información de perfiles de usuario como CRM, CPEX o P3P.

References

[1] Web de Amazon. www.amazon.co.uk, 2002.

[2] M. Bieber, H. Oinas-Kukkonen, and V. Balasubramanian. Hypertext Functionality. *ACM Computing Surveys*, 31(4), 12 1999.

[3] C. Cachero, I. Garrigós, and J. Gómez. Personalización de Aplicaciones en OO-H. In *Quintas Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software (IDEAS'02)*, 04 2002.

[4] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites WWW9 Conference. In *First ICSE Workshop on Web Engineering, International Conference on Software Engineering*, 05 2000.

[5] P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: a Survey. *ACM Computing Surveys*, 2000.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison Wesley, 1995.

[7] A. Ginige and S. Murugesan. Web Engineering: an Introduction. *IEEE Multimedia Special Issue on Web Engineering*, pages 14–18, 04 2001.

[8] J. Gómez, C. Cachero, and O. Pastor. Conceptual Modelling of Device-Independent Web Applications. *IEEE Multimedia Special Issue on Web Engineering*, pages 26–39, 04 2001.

[9] G. Kappel, W. Retschitzegger, and W. Schwinger. Modeling Customizable Web Applications - A Requirement's Perspective. In *2000 Kioto International Conference on Digital Libraries*, 11 2000.

[10] N. Koch, A. Kraus, and R. Hennicker. The Authoring Process of the UML-based Web Engineering Approach. In *Proceedings of the 1st International Workshop on Web-Oriented Software Technology*, 05 2001.

[11] B. Mobasher, R. Cooley, and J. Srivastava. Creating Adaptive Web Sites Through Usage-Based Clustering of URL's. In *IEEE Workshop on Knowledge and Data Engineering Exchange*, 11 1999.

[12] J. Pavón. Personalización de servicios en la Web. ZOCO 2001. http://tdg.lsi.us.es/ zoco/res/ppt/pavon.zip, 2001.

[13] W. Retschitzegger and W. Schwinger. Towards Modeling of DataWeb Applications - A Requirement's Perspective. In *Proceedings of the American Conference on Information Systems AMCIS 2000*, volume 1, pages 149–155, 08 2000.

[14] Daniel Schwabe and Gustavo Rossi. A Conference Review System with OOHDM. In *First International Workshop on Web-Oriented Software Technology*, 05 2001.

[15] Olga De Troyer and Sven Casteleyn. The Conference Review System with WSDM. In *First International Workshop on Web-Oriented Software Technology*, 05 2001.

[16] eXtensible Markup Language. http://www.w3.org/XML/, 2000.