Tratamiento de la Personalización Dinámica en Modelos Conceptuales de Aplicaciones Web*

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

IWAD Group
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante. SPAIN
{igarrigos,jgomez}@dlsi.ua.es
http://www.dlsi.ua.es/iwad/

Abstract Las aproximaciones de Modelado Conceptual para la web necesitan extensiones para especificar propiedades de personalización dinámica para diseñar aplicaciones web más potentes. Las propuestas actuales proveen técnicas para soportar personalización dinámica, usualmente enfocadas en detalles de implementación. Este artículo presenta una extensión de la aproximación de modelado conceptual OO-H para abordar los detalles asociados con el diseño y la especificación de la personalización dinámica. Describimos cómo los diagramas convencionales de navegación y presentación se ven afectados por propiedades de personalización. Para modelar la parte variable de la lógica de interfaz OO-H tiene una arquitectura de personalización que se apoya en un motor de reglas. Las reglas se definen basándose en un *Modelo de Usuario* y un *Modelo de Referencia*. OO-H provee dos tipos de reglas: reglas de adquisición, en las que se recoge la información necesaria, y reglas de personalización que se aplican a los niveles de navegación y presentación, y se ven reflejadas en sus correspondientes modelos conceptuales. De esta manera, la lógica de interfaz de una aplicación web puede verse como una composición de una parte estable y otra parte variable, donde la parte variable (expresada en XML) se interpreta en tiempo de ejecución. El principal beneficio es que esta especificación puede modificarse sin recompilar el resto de los módulos de la aplicación.
Keywords Ingeniería Web, Modelado Conceptual, Personalización, XML

1 Introducción

Las aproximaciones de Ingeniería Web actuales [5] ayudan a los diseñadores a hacer más sencilla la comprensión, desarrollo, evolución y mantenimiento de aplicaciones web. Estos métodos están basados en nuevos constructores y vistas hipermediales [8, 11, 2, 12] que abordan el problema de la navegación/presentación del usuario a través del espacio de información. Sin embargo, las aproximaciones que tratan algún tipo de personalización varían ampliamente. En este contexto,

* Este artículo ha recibido la ayuda parcial del Ministerio Español de Ciencia y Tecnología, proyecto número TIC2001-3530-C02-02

el principal problema es la falta de constructores de modelado conceptual para especificar personalización dinámica. Nosotros pensamos que se necesitan nuevas técnicas para extender los metamodelos con aspectos de personalización preservando previos modelos conceptuales si queremos aprovechar esfuerzos anteriores de modelado. Además, necesitamos tratar la personalización por medio de un proceso de externalización donde las reglas de personalización pueden ser interpretadas en tiempo de ejecución. De este modo, podemos proveer un tratamiento flexible de la personalización que es independiente de la tecnología.

Este artículo presenta como el método OO-H *(Object Oriented Hypermedia)* [7, 6] se extiende para soportar personalización dinámica. Describimos cómo los diagramas convencionales de navegación y presentación se ven influenciados por propiedades de personalización. Estas propiedades se capturan en forma de ficheros externos escritos en XML, que representan las reglas. Estas reglas, que forman la parte variable de la lógica de interfaz, serán tratadas en tiempo de ejecución por un motor incluido en la arquitectura de ejecución. El soporte de esta arquitectura se consigue con dos modelos: un *modelo de referencia*, que registra la actividad del usuario en el sistema y un *modelo de usuario* que recoge la información necesaria para personalizar. De este modo, la lógica de interfaz de una aplicación web se puede ver como una composición de una parte estable y una parte variable, donde la parte variable (expresada en XML) es interpretada en tiempo de ejecución.

El artículo está estructurado de la siguiente manera: la sección 2 presenta trabajo relacionado en el campo de la personalización dinámica. La sección 3 muestra los elementos que soportan la personalización en OO-H. La subsección 3.1 presenta la arquitectura de ejecución subyacente en las aplicaciones web de OO-H. En la subsección 3.2 se describe cómo se captura el conocimiento que el sistema tiene sobre el usuario por medio de un modelo de usuario. La subsección 3.3 presenta cómo la estrategia de modelado de la personalización se define por medio de un conjunto de estructuras de información expresadas en un modelo de referencia. A continuación, en la sección 4 se describe, por medio de un ejemplo, como se consigue soportar la personalización en OO-H. Finalmente, la sección 5 presenta las conclusiones y los trabajos futuros.

2 Estado del Arte

Con respecto a las aproximaciones de Ingeniería Web, algunas proveen soporte para la personalización en base a roles, como OOHDM [11] o WSDM [12]. En OOHDM el modelo de interfaz abstracto es el resultado de la especificación de los objetos de interfaz que percibirá el usuario. OOHDM utiliza *Vistas Abstractas de Datos* (*Abstract Data Views, ADVs*) para modelar los aspectos estáticos de la interfaz de usuario [9] mientras que los aspectos dinámicos de la interfaz de usuario se modelan con una técnica basada en diagramas de estado (*StateCharts*) [3]. El modelado y diseño en un marco conceptual permite un mejor entendimiento

de los mecanismos utilizados y el descubrimiento de características comunes que permiten el reuso de patrones, componentes, algoritmos o incluso subsistemas [11]. En la aproximación WSDM [12] se definen diferentes perspectivas para las clases de usuarios; los distintos tipos de usuarios pueden ver la misma información y navegar a través de la información de diferentes maneras.

Otras aproximaciones proveen marcos que complementan al modelo conceptual y soportan características de adaptación, adaptividad e incluso de proactividad, como W3I3 [2] o UWE [8]. W3I3 [2] incluye mecanismos para externalizar políticas y soportar su cambio una vez que haya sido implantada la aplicación. Sin embargo, estos mecanismos se implementan como triggers de bases de datos y están enfocados en el diseño de aplicaciones web de datos intensivos. UWE [8] se centra en la especificación de aplicaciones adaptivas. Insiste en las características de personalización, como la definición de un modelo de usuario o un conjunto de características de navegación adaptivas que dependen de preferencias, conocimiento o tareas que debe ejecutar el usuario. Desde nuestro punto de vista esta es la propuesta más completa. Provee un marco teórico formal para soportar la personalización dinámica. Sin embargo, el principal problema es la inflexibilidad de los modelos conceptuales de UWE. Esto supone que cualquier cambio que el diseñador quiera hacer debe estar soportado en el marco de trabajo de UWE.

Por último, hay un gran número de herramientas comerciales (e.g. ILog JRules, LikeMinds, WebSphere, Rainbow..) que facilitan el uso de técnicas y estrategias de personalización y dan soporte a muchas aplicaciones web personalizadas. Estas herramientas están orientadas a la implementación de estrategias de personalización. El principal problema es el bajo nivel de abstracción que causa problemas de reuso y dificulta el mantenimiento y la escalabilidad de las aplicaciones personalizadas resultantes. A continuación vamos a presentar como OO-H resuelve algunas de estas limitaciones.

3 Soporte de la Personalización en OO-H

3.1 Arquitectura de la Personalización

La arquitectura de OO-H se ha extendido para soportar personalización dinámica (ver Fig. 1). Más concretamente, las propiedades de personalización se capturan en el nivel de navegación/presentación y se ven reflejadas en sus correspondientes modelos conceptuales (DAN, CLD) por medio de un conjunto de reglas de asociación. De esta forma el diseño y generación de la lógica de navegación puede especificarse en dos partes: una parte estable, que es independiente de las propiedades de personalización, y una parte variable, que soporta el tratamiento de estas reglas. Finalmente, un motor de reglas provee el contexto para interpretar las reglas generadas en tiempo de ejecución.

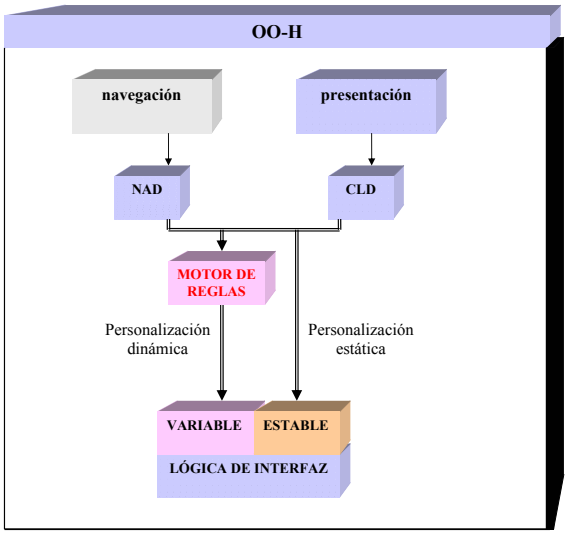

Figure1. *Arquitectura de OO-H*

Modelar por separado las partes variable y estable de la aplicación desde las primeras fases del ciclo de desarrollo del software, facilita el tratamiento de la naturaleza dinámica de las aplicaciones web, así como el de 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 [10]. El soporte de esta arquitectura se consigue a partir de un *Modelo de Referencia* que permite capturar las propiedades relevantes de personalización. Además, se debe definir un *Modelo de Usuario* para soportar los requisitos de personalización. En la siguiente sección comenzaremos presentando el *Modelo de Usuario* y en la sección 3.3 presentaremos el *Modelo de Referencia*.

3.2 El Modelo de Usuario

El *modelo de usuario* es una parte importante de un sistema adaptivo hipermedial ya que la información almacenada en el modelo de usuario permite modificar o ajustar información (contenido adaptivo), proveer al usuario con soporte de navegación (navegación adaptiva) o individualizar capas (presentación adaptiva). Se define como la representación de las creencias o conocimiento que el sistema tiene acerca del usuario. Un modelo de usuario está constituido por descripciones de lo que se considera relevante sobre el conocimiento actual y/o aptitudes del usuario, proveyendo información para el entorno del sistema para adaptarse al usuario individual [8].

En OO-H el modelo de usuario se captura por medio de un Diagrama de Clases que complementa al Modelo Conceptual. Captura información acerca de las características que el sistema cree que tiene el usuario. Estas características

pueden ser preferencias o intereses, conocimiento general, experiencia, etc. Un modelo de usuario no necesita ser completamente preciso, y de hecho usualmente se restringe a aproximaciones imprecisas. Sin embargo, incluso un modelo de usuario incompleto puede ser útil [8]. La información que debería contener este modelo depende de los requisitos de personalización que queramos soportar.

En OO-H el modelo de usuario se centra en los conceptos de usuario y rol, al igual que en otras aproximaciones hipermediales [2]. Para proveer de vistas personalizadas al usuario, OO-H provee un modelo de usuario básico. Este modelo de usuario se construye alrededor de una clase llamada *Usuario*, que provee la información y comportamiento que debe ser heredado por cada <<*actor*>> del sistema. Un usuario puede no tener asociado un rol, en este caso ella/él sería tratado como un *usuario anónimo*. Además, un usuario sólo puede tener asociado un rol al mismo tiempo. Este modelo puede enriquecerse para soportar la política de personalización deseada añadiendo atributos, métodos o enlaces desde la clase Usuario al resto de las clases del dominio o al *Modelo de Referencia OO-H*, que se presenta en la siguiente sección. En la Fig. 2 podemos ver un posible modelo de usuario.

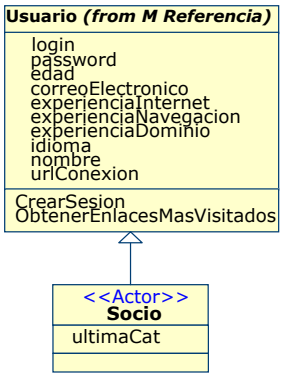

Figure2. *Modelo de Usuario*

Este modelo de usuario surge de la conexión de la clase *Usuario*, del modelo de Referencia, con la clase existente *Socio*, del modelo Conceptual, y lo trata como un rol. Esta clase *Socio* proviene del modelado conceptual de un video club en internet, sistema que utilizaremos en el ejemplo de modelado que veremos en la sección 4.

La estrategia de modelado de personalización debe ser definida dependiendo de un conjunto de estructuras de información. Esta información es almacenada en OO-H en un repositorio que contiene el conjunto inicial de elementos básicos de

información en el que se puede establecer la política de personalización deseada. Esto es lo que se llama el modelo de Referencia que se explica a continuación.

3.3 Modelo de Referencia

Este respositorio contiene el conjunto inicial de elementos básicos de información sobre los que se puede establecer la política de personalización deseada. El principal beneficio de utilizar este modelo de Referencia es que 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.

La estrategia de modelado de la personalización se define en función de una serie de estructuras de información. Esta información se almacena en OO-H en un repositorio que se presentó en [1]. El actual modelo de Referencia OO-H estructura la información para modelar la personalización en tres partes: perfiles de usuario, información de contexto y reglas de asociación. Para los propósitos de este artículo prestaremos más atención a la parte que modela la personalización en base a las **Reglas de asociación** (en la Fig. 3 podemos ver esa parte del modelo de Referencia). OO-H incorpora un mecanismo externo para modelar la personalización de las aplicaciones adaptivas y proactivas en forma de reglas de asociación. Una ventaja de usar estas reglas es que están definidas en ficheros externos y pueden modificarse de una manera independiente del resto de la aplicación. Este tipo de reglas permiten capturar las propiedades de personalización embebidas directamente en los modelos de OO-H.

Estas reglas se han dividido en tres tipos:

- **Reglas de Adquisición**: en las que se recoge la información necesaria para la personalización.
- **Reglas de Personalización**: soportan la acción de personalización. Como hemos visto en la Fig. 3, este tipo de reglas se categoriza a su vez, basándose en lo que queremos personalizar, en:
  - **Reglas de Contenido**: usando estas reglas podemos *Seleccionar*, *Ordenar* o *Modificar* la información de contenido de la aplicación.
  - **Reglas de Navegación**: estas reglas afectan al modo de navegación. Con este tipo de regla podemos *Añadir, Borrar, Activar u Ordenar* cualquier enlace en la navegación del usuario.
  - **Reglas de Presentación**: esta regla afecta a aspectos de presentación. Las posibles acciones que pueden asociarse a este tipo de regla son: *EstablecerPlantillaCSS o ModificarPresentacion*, dependiendo de si queremos usar una plantilla predefinida de presentación o queremos modificar un valor específico de la presentación. Tenemos un conjunto de plantillas CSS, que pueden asociarse a casos específicos (como para personas con minusvalías) capturados en el *Modelo de Referencia* y una *Clase ModificarPresentación* con características de presentación (ver Fig 3). Además,

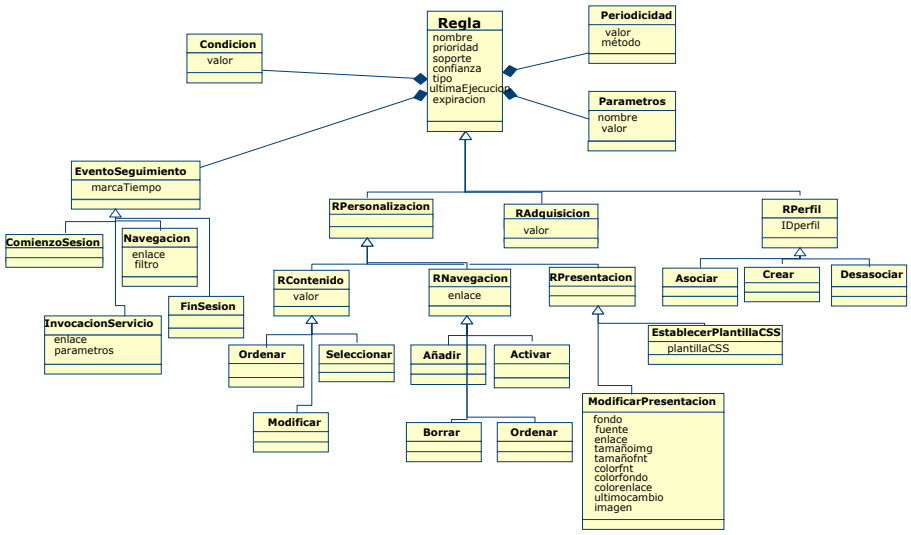

Figure3. *Parte del Metamodelo de OO-H*

en el *Modelo de Usuario* podemos completar la Clase Presentación con las características diseñadas para un modelo conceptual específico. Las reglas permiten la selección de una plantilla predefinida o la modificación de características específicas de la Clase Presentación.

– **Reglas de Perfil**: estas son reglas de manejo de perfiles. Con este tipo de reglas podemos *Asociar o Desasociar* comportamiento específico a perfiles de usuario, o bien podemos *Crear* un nuevo perfil.

OO-H soporta la especificación de estas reglas por medio de un esquema XML. Hay que destacar que la arquitectura de ejecución de las aplicaciones generadas desde los modelos OO-H permite modificar y reprocesar este esquema sin recompilar el resto de los módulos. En el esquema los diferentes elementos XML corresponden a clases/atributos del marco presentado en la Fig. 3. Todas las reglas son reglas ECA (Evento-Condición-Acción) [4]. Además, en las *reglas de Adquisición* se puede establecer una periodicidad de la regla. Esta periodicidad especifica el intervalo en el que la aplicación, de una forma automática, tiene que comprobar esa regla. En el contexto de la etiqueta de periodicidad, OO-H define un valor especial, "*ooh:always*", que indica que el cumplimiento de las condiciones de activación debe ser comprobado continuamente. La acción que puede implementar una regla varía dependiendo del tipo de regla, como hemos visto. Vamos a ver cómo se hacen efectivas las reglas en los modelos de navegación de OO-H.

En la siguiente sección se presenta un ejemplo de modelado de personalización del Contenido. El sistema a modelar es un video club en Internet que maneja clientes, películas y alquileres.

4 Ejemplo de Modelado

El *Modelo de Usuario* y el *Modelo de Referencia*, junto con las *Reglas de Asociación*, nos permiten personalizar el contenido, la navegación y la presentación de una aplicación. En el contexto del sistema mencionado (video club en Internet) vamos a considerar el siguiente requisito:

- Cuando se alquila una película, mostrar recomendaciones de películas con la misma categoría *(Personalización del Contenido)*.

El primer paso para soportar este requisito es definir el *Modelo de Usuario*. El modelo de usuario que soporta este requisito fue mostrado en la sección 3.2, en la Fig. 2. Además, para especificar la personalización hemos tenido que definir cómo soportar el requisito mencionado. Tenemos que distinguir entre la adquisición y la personalización de los datos correspondientes. El soporte de este requisito se detalla en dos fases:

- *Proceso de Modelado y Recogida*, donde se muestra cómo se introduce el requisito en el diagrama DAN/CLD.
- *Especificación XML*, donde se muestra una especificación XML que soporta el requisito.

Finalmente se mostrará la pantalla o conjunto de pantallas que resulta de soporta el requisito definido.

4.1 Personalización del Contenido

El requisito definido requiere la personalización del contenido. En este caso el contenido a personalizar es el conjunto de películas que vamos a mostrar como recomendaciones, que variará según la película que haya alquilado el usuario. Vamos a ver primero la regla de adquisición de la información necesaria para soportar este requisito, que en este caso es la categoría de la película que ha alquilado el usuario.

Regla de Adquisición (1a):

PROCESO DE MODELADO Y RECOGIDA

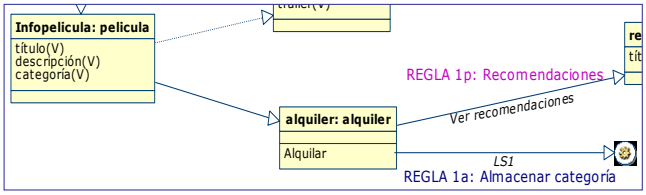

Figure4. *Adquisición: regla 1a*

La figura 4 muestra un trozo de diagrama DAN en el que se introduce la regla de adquisición que modela este requisito que se llama *Almacenar Categoría*. Podemos ver que esta regla se asocia a un enlace de invocación de servicio que corresponde a la invocación del método *Alquilar*, esto es así ya que, cómo se deduce del requisito, necesitamos almacenar la categoría de la película alquilada para después poder mostrar las recomendaciones de películas que tengan esa misma categoría (la de la película que acaba de alquilar el cliente).

ESPECIFICACIÓN XML

El resultado de este proceso de adquisición se almacena como una especificación XML que será interpretada por un motor de reglas en tiempo de ejecución.

```
<TPersonalizacion>
    ...
    <perfil nombre="ooh:all" condicion="">
        <regla tipo="adquisicion" nombre="AlmacenarCategoria" soporte="20"
            confianza="100" prioridad="Media" activacion="12/02/03"
            expiracion="12/02/04" ultimaEjecucion="23/09/03">
            <evento tipo="InvocacionServicio" enlace="Alquilar"/>
            <accion valor="sesion.socio.ultimacat=alquilar.pelicula.categoria" />
        </regla>
    </perfil>
    ...
</TPersonalizacion>
```

En el esquema los diferentes elementos XML corresponden a clases/atributos del marco presentado en la Fig. 3. En primer lugar, observamos cómo la regla se define en el contexto de un perfil. En este caso, 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. Otros atributos asociados a las reglas son:

- Nombre de la regla
- Tipo de regla: En este caso es una regla de adquisición de información (necesaria para después personalizar).
- Soporte: Indica el porcentaje de veces que se ha activado la regla en nuestra aplicación.
- Confianza: Indica el porcentaje de aciertos en la ejecución de la regla.
- Prioridad: Indica la importancia de la regla en el sistema en caso de conflicto o condiciones de carrera.

Esta especificación describe cómo esta regla de adquisición afecta a todos los perfiles y tiene prioridad media. La información que necesitamos adquirir es la categoría de la última película alquilada. Este tipo de adquisición es implícita y se almacena cuando un socio alquila una película en el atributo del Modelo de Usuario: "sesion.socio.ultimacat". A continuación vamos a ver la regla de personalización que modela este requisito.

Regla de Personalización (1p):

PROCESO DE MODELADO Y RECOGIDA

La figura 5 muestra la parte del diagrama DAN en el que se introduce la regla de personalización que modela este requisito llamada *Recomendaciones*. Esta regla está asociada al enlace *Ver Recomendaciones*, esto implica que cuando se activa este enlace se dispare la regla de personalización que mostrará las recomendaciones que tengan la misma categoría que la película que acaba de alquilar el usuario.

Figure5. *Personalización: regla 1p*

ESPECIFICACIÓN XML

En este caso, tenemos personalización del contenido, porque tenemos que mostrar u ocultar información específica. En la especificación XML podemos asociar parámetros a una regla para simplificar la notación. En este caso, como parámetro tenemos la categoría de la última película alquilada, que es la información que se obtuvo mediante la regla de adquisición y se almacenó en el atributo del modelo de usuario *sesion.socio.ultimacat*. El evento de personalización indica que tiene que estar activo el enlace "Ver Recomendaciones". Si estas condiciones se cumplen, se ejecutará la acción: se muestran las películas con las condiciones impuestas.

```
<TPersonalizacion>
    ...
    <perfil nombre="ooh:all" condicion="">
        <regla tipo="personalizacion:contenido" nonmbre="Recomendaciones" soporte="20"
            confianza="100" prioridad="Media" activacion="12/02/03"
            expiracion="12/02/04" ultimaEjecucion="23/09/03">
            <parametros>
            <parametro nombre="catpelicula" valor="sesion.socio.ultimacat"/>
            </parametros>
            <evento tipo="Navegacion" enlace="Ver Recomendaciones"/>
            <condicion valor="pelicula.categoria=catpelicula"/>
            <accion tipo="Seleccionar" valor="pelicula.categoria"/>
        </regla>
    </perfil>
    ...
</TPersonalizacion>
```

REPRESENTACIÓN FINAL

Una vez visto el modelado de este requisito vamos a ver el escenario final resultante. En la figura 6 se muestra la pantalla resultado de aplicar las reglas

de adquisición y personalización del contenido; en esta imagen podemos ver la pantalla resultante de haber alquilado una película. Vemos el enlace *Ver Recomendaciones* el cual, al ser activado, muestra recomendaciones de películas de la misma categoría que la película que acabamos de alquilar.

Figure6. *Escenario Final*

5 Conclusiones y trabajos futuros

Los métodos de Ingeniería Web han de proveer un proceso de desarrollo del software bien definido mediante el que la comunidad de ingenieros software puedan diseñar adecuadamente aplicaciones potentes basadas en web de una manera sistemática. Nuestro propósito ha sido tratar estos problemas en el contexto de la aproximación de modelado conceptual OO-H que ha sido probada con éxito para el diseño de aplicaciones web. Nos centramos en cómo capturar correctamente las características particulares asociadas al diseño de personalización dinámica. Para conseguir este objetivo, OO-H añade reglas de adquisición y de personalización, que definen la semántica adecuada para capturar y representar la funcionalidad específica de la personalización dinámica. De este modo, los modelos de navegación y presentación pueden compilarse para obtener una especificación XML que represente la personalización dinámica deseada. La aplicación web final se puede ver como una composición de una parte estable y una parte variable, donde la parte variable es interpretada por un motor de reglas para dar soporte de personalización. El principal beneficio es que la especificación de personalización puede modificarse sin necesidad de recompilar el resto de los módulos

de la aplicación. Otras contribuciones relevantes de este artículo son las siguientes: (1) un modelo de usuario que describe cómo se captura el conocimiento que tiene el sistema del usuario. (2) un modelo de referencia que permite extender el metamodelo OO-H mediante un conjunto específico de estructuras de información. OO-H todavía está definiendo y catalogando un conjunto de patrones de personalización de la navegación y de la presentación suficientemente genérico como para garantizar la reusabilidad de la aplicación.

Además, esta forma de soportar personalización dinámica está siendo implementada en el entorno CASE OO-H para obtener resultados empíricos de la aplicabilidad de este trabajo.

Otro trabajo que se tiene previsto realizar es la comprobación de consistencia de las reglas especificadas (reglas redundantes, reglas en conflicto, sentencias if innecesarias...etc) y la comprobación de terminación y completitud de las reglas.

References

[1] 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.

[2] 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.

[3] Harel D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3), 1987.

[4] U. Dayal. Active Database Management Systems. In *Proc. 3rd Int. Conference on Data and Knowledge Bases*, pages 150–169, 1988.

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

[6] J. Gómez, C. Cachero, and O. Pastor. Extending a Conceptual Modelling Approach to Web Application Design. In 12^{th} *International Conference on Advanced Information Systems (CAiSE'00)*, volume 1789, pages 79–93. Springer-Verlag. Lecture Notes in Computer Science, 06 2000.

[7] 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.

[8] 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.

[9] Carneiro L., Cowan D., and Lucena C. Introducing ADV Charts: A Graphical Specification of Abstract Data Views. In *Proceedings of CASCON'93*, 1993.

[10] 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.

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

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