OOWS: Un Método de Producción de Software en Ambientes Web*

Joan Fons, Oscar Pastor, Pedro Valderas y Marta Ruiz

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
46022 Valencia, Spain
{jjfons, opastor, pvalderas, mruiz}@dsic.upv.es

Abstract. Este artículo presenta una aproximación para el modelado orientado a objetos de soluciones web que proporciona mecanismos para la especificación de sistemas de información dinámicos hipermediales y de comercio electrónico. Se propone una extensión de un método OO para la generación automática de código usando modelos conceptuales (OO-Method) que permite capturar y representar la semántica navegacional y de presentación de información usando primitivas de abstracción de alto nivel. Se define un proceso guiado de construcción de una solución web completa y se aplica la aproximación planteada a un caso de estudio sobre una aplicación para el comercio electrónico.

Introducción

Hoy en día, con la rápida expansión de Internet y los avances en el área de las tecnologías web han aparecido un nuevo tipo de aplicaciones en estos entornos, y son cada vez más complejas y dinámicas. Además, debido al acelerado crecimiento y la alta competitividad de las actividades comerciales en la Red, estos sistemas son construidos en periodos temporales muy cortos, sin el apoyo de herramientas de trabajo adecuadas y utilizando soluciones ad-hoc, lo que está llevando a construir sistemas software de baja calidad y de difícil mantenimiento y evolución.

En los últimos años han surgido gran cantidad de aproximaciones metodológicas que intentan ayudar en la sistematización de la construcción de soluciones en ambientes web, proporcionando mecanismos de abstracción que faciliten el desarrollo de estos sistemas. Además, se están intentando definir marcos de trabajo integrados que proporcionen herramientas adecuadas para dar soporte a la construcción de estos sistemas en todas sus fases. Pero actualmente no existe ningún método totalmente establecido.

Existen dos tendencias claras. Unas aproximaciones se basan en extender iniciativas orientadas al diseño hipermedial (navegacional), introduciendo expresividad para dotar de dinamismo a los sistemas. Estas aproximaciones aparecieron hacia el princi-

* Esta investigación está soportada por el Programa CYTED, en el proyecto VII.18, WEST y el Proyecto CICYT del programa FEDER, con ref. TIC 1FD97-1102

pio o mitad de la década de los 90, con el objetivo de construir aplicaciones hipermediales donde se unía el concepto de navegación con la multimedia, en sistemas claramente estáticos (sin funcionalidad). Es por esto que la mayoría de estas aproximaciones están basadas en el Modelo Relacional clásico, o bien en extensiones de éste. Algunos ejemplos destacables de estas iniciativas son OOHDM [22] (actualización OO de HDM [9]), WebML [6], ADM [14], AutoWeb [16] y RMM [11]. El otro grupo de aproximaciones se basan en la idea de extender los métodos de desarrollo orientados a aplicaciones dinámicas (con funcionalidad), que podríamos llamar "convencionales", tratando de introducir la semántica de la hipermedia como característica inherente a este nuevo tipo de sistemas software. Este tipo aproximaciones (por lo general, más recientes) tratan de introducir características navegacionales al modelo OO. En este grupo podemos encontrar los métodos UWE [4], WSDM [7], EORM [13], OOH [10] y OO-Method [17].

Sin embargo, en estas propuestas, las características hipermediales y las propiedades funcionales son tratadas habitualmente por separado, dificultando el problema de desarrollar una aplicación web en un marco de trabajo unificado. En la práctica, estos métodos proporcionan una solución parcial, bien porque se centran en captar las características navegacionales (en detrimento de la especificación funcional del sistema), o bien en captar las características más convencionales (definición de clases y operaciones para expresar la funcionalidad), sin tener en cuenta la semántica navegacional de los sistemas.

Detrás de estas soluciones parciales está empezando a ser ampliamente aceptado que los sitios web están evolucionando de simples repositorios de información hipermedia hacia complejas aplicaciones hipermediales distribuídas, normalmente conocidas como "aplicaciones web" [3]. Además, la complejidad de los sistemas aumenta debido al gran tamaño y la cantidad de potenciales tipos diferentes de usuarios con los que pueden interactuar. Por tanto, se deben proporcionar mecanismos que faciliten y guíen la construcción de soluciones y favorezcan el reuso de soluciones ya probadas.

Nuestra propuesta proporciona una contribución en este contexto: empezando por afirmar que el modelado conceptual es necesario para desarrollar aplicaciones web correctas y de calidad, introducimos una aproximación diferente que se centra en crear un entrono de trabajo integrado, donde se aborda tanto la perspectiva funcional del sistema, como la perspectiva navegacional y la de presentación de la información. Todas estas perspectivas unidas pueden ser usadas como entrada para ambientes de generación automática de código. Para poner en práctica estas ideas, se ha diseñado e implementado un proceso de producción de software, donde se definen unas guías para aplicar técnicas de modelado conceptual para el desarrollo de aplicaciones web: integrar las tres actividades que tradicionalmente se llevan a cabo por separado. Una primera donde se modelan las operaciones, una segunda donde se expresa el concepto de navegación y una tercera donde se plasman los requisitos de presentación de información usando primitivas básicas de presentación a nivel de modelado conceptual. Así, en nuestra aproximación, la fase de modelado conceptual se considera como una única fase. Usando lo que podríamos llamar una aproximación de modelado conceptual OO, se introduce la expresividad necesaria en el modelo para especificar correctamente los requisitos navegacionales y las características de presentación. Toda esta información es utilizada para proporcionar una guía metodológica precisa para ir del espacio del problema (modelado conceptual) al espacio de la solución (representado

por el producto software final), soportado por un proceso de generación automática de código.

Este artículo está organizado en 4 secciones. La sección 2 describe el método de producción de soluciones web *OOWS*, describiendo el proceso de desarrollo, el modelo de navegación y el modelo de presentación. En la sección 3 se aplica el método a un caso de estudio, y en la sección 4 se presentan las conclusiones y los trabajos futuros.

OOWS: Un método de desarrollo de aplicaciones web

Nuestra intención es definir un método de desarrollo que permita especificar sistemas software para ambientes web, extendiendo un método OO existente. Este tipo de aplicaciones tienen una base común con las aplicaciones software tradicionales: la funcionalidad del sistema y la interacción con los usuarios. Sin embargo, introducen nuevas características *navegacionales* que deben ser capturadas para representar de una manera más precisa y aproximada el sistema.

El método tomado como la base para definir esta aproximación es OO-Method [19]. Este método capta las propiedades funcionales del sistema que se consideran relevantes para construir una especificación textual OO y formal de manera automática. Esta especificación formal OO constituye un repositorio de información de alto nivel del sistema que será utilizado como entrada a un *compilador (automático) de modelos conceptuales (model compiler)*. La definición de un modelo de ejecución junto con una estrategia basada en patrones de traducción (de la especificación a la implementación), hacen posible la construcción de una implementación operacional, generando un prototipo del sistema completo (incluyendo las características estáticas y dinámicas) en la plataforma destino del sistema. En el contexto del proyecto OO-Method se han dirigido muchos esfuerzos hacia el desarrollo de nuevos modelos para enriquecer el Método de Producción de Software Orientado a Objetos con la expresividad necesaria para especificar las características navegacionales. Esta extensión del método OO-Method con capacidades navegacionales y de presentación es lo que llamamos OOWS [18][20] (Método de Producción de Soluciones Web Orientadas a Objeto).

Proceso de desarrollo de una aplicación web

En este apartado se define el proceso o guía metodológica a seguir para definir y posteriormente obtener un sistema software para un entorno web. Siguiendo la aproximación OO-Method, la especificación de los requisitos de usuario debe realizarse en la fase de modelado conceptual. Para modelar la navegación asociada al sistema deseado se propone un proceso de desarrollo de soluciones-web con dos pasos principales: Especificación del Problema y Desarrollo de la Solución.

En la fase de **Especificación del Problema** se deben capturar las peculiaridades y el comportamiento que debe ofrecer el sistema para satisfacer los requisitos de usuario identificados. En este paso se incluye el conjunto de requisitos usando una aproximación de Casos de Uso [12] y posteriormente las actividades de modelado conceptual del sistema. En el modelado conceptual, las abstracciones que se derivan

del problema son especificadas en términos de clases y de su estructura, comportamiento y funcionalidad, construyendo los siguientes modelos: de Objetos, Dinámico, Funcional, Navegacional y de Presentación. Éstos describen la sociedad de objetos desde cinco puntos de vista diferentes usando un marco de trabajo OO bien definido:

- El Modelo de Objetos define la estructura y las relaciones estáticas entre clases identificadas en el dominio del problema.
- En el Modelo Dinámico se describen las posibles secuencias de servicios y los aspectos relacionados con la interacción entre objetos.
- El Modelo Funcional captura la semántica asociada a los cambios de estado entre los objetos motivados por la ocurrencia de eventos o servicios.
- El Modelo de Navegación define la semántica navegacional asociada a las clases de los objetos del modelo. Es en este modelo donde se explicita la navegación permitida en la aplicación para cada agente del sistema.
- El Modelo de Presentación captura los requisitos básicos de presentación de información, orientado hacia ambientes web. Está fuertemente basado en el modelo de navegación y permite definir, de una manera abstracta la estructura lógica de presentación de los objetos navegacionales en la interfaz de usuario.

En esta fase se realiza un estudio de los tipos de usuarios que pueden interactuar con el sistema, indicando qué visibilidad sobre el sistema tendrán (qué atributos y qué operaciones podrán ver y/o activar), cómo se podrán conectar (requerirán o no identificación), y se organizarán en jerarquías de especialización [8] para potenciar el reuso en la especificación del sistema, facilitando así la tarea de modelado.

En la fase de **Desarrollo de la Solución** se propone una estrategia de generación de código basada en componentes para integrar la solución propuesta en ambientes web. En esta etapa se obtendrá una aplicación web, con una funcionalidad equivalente a la especificación inicial según una visión operativa. Esta fase se desarrolla de manera totalmente automática por un compilador de modelos conceptuales (*model compiler*) que, en función de unos patrones arquitectónicos y dependiendo de la plataforma destino, construye un sistema software que recoge los requisitos de la aplicación modelada, siguiendo uno de los principales objetivos de la metodología propuesta por OO-Method. Esta estrategia, además, facilita las tareas de mantenimiento y evolución, ya que la generación automática basada en patrones se realiza utilizando soluciones previamente probadas y validadas. Un cambio en el sistema (evolución), implicará una modificación en el modelo conceptual y una regeneración automática del sistema. Además, esta filosofía nos permite obtener de una manera más rápida aplicaciones finales de calidad, evitando entre otras, la fase de pruebas (*testing*) del sistema.

A continuación se presentan las extensiones navegacionales y de presentación de información que se introducen en la propuesta OO-Method, a lo que hemos llamado la aproximación OOWS.

El Modelo de Navegación de OOWS

En esta sección se presentan las primitivas de modelado conceptual para extender un método OO con la intención de capturar la semántica navegacional de una aplicación

web. En el método OOWS, se incorpora un nuevo modelo en la fase de modelado conceptual que recoge las características navegacionales: el *Modelo de Navegación*. Su objetivo es definir cómo se le proporcionará a cada usuario del sistema el acceso a la información y la funcionalidad que le es relevante para llevar a cabo su tarea dentro del sistema y qué secuencias de caminos deberán seguir para conseguirlo.

Actualmente, y debido al gran auge de las aplicaciones de carácter comercial en Internet (*e-commerce*), es de vital importancia que el acceso al sistema facilite la interacción con el sistema e incluso se le personalice. Un ejemplo ilustrativo es la tienda virtual Amazon.com [2], en donde, si se conecta un usuario registrado, el sistema mantiene información sobre sus preferencias, sus compras anteriores, sus productos más deseados, etc., proponiéndole nuevas compras que puedan ser de su interés. Esto nos lleva a tener en cuenta también aspectos referentes a personalización a nivel de usuario [21][1][5].

Pero existe actualmente una gran controversia en lo que se refiere al concepto de navegación: ¿qué es la navegación?. ¿Cómo la debemos captar y expresar? ¿Se debe definir a nivel de modelado conceptual o a nivel de implementación?. Existen muchos autores que giran en torno a la misma idea, pero sin proporcionar una definición que sea aceptada definitivamente. Algunos llegan incluso a confundir conceptos, como por ejemplo tratar parte de la funcionalidad del sistema como requisito de navegación (debido a que puede haber una fuerte interrelación). Nosotros proponemos una definición de navegación lo suficientemente genérica que pueda ser aceptada por cualquier método o propuesta. Para ello revisemos las características básicas y novedosas que incorpora una aplicación web. Cuando un usuario se conecta a una aplicación web, ésta le proporciona una visión del sistema software formada por un conjunto de *nodos enlazados* entre sí definiendo los posibles caminos que el usuario puede tomar. Esta estructura se puede organizar en un grafo dirigido donde los **nodos** representan los puntos de interacción con el usuario, proporcionándole un conjunto de datos y/o servicios (cohesivos) con los que el usuario puede interactuar; los **arcos** representan una alcanzabilidad entre nodos, indicando los posibles "siguientes" nodos (o caminos) que se puede alcanzar o seguir. Según esta visión, la navegación será todo "cambio de nodo provocado al activar un enlace de navegación". De esta manera podemos definir la navegación a nivel de modelado conceptual y se elimina la problemática de definirla en términos del espacio de la solución (número de peticiones al servidor, paginación (*scrolling*) de datos, ocultación parcial de información en la misma página, etc.).

En la aproximación OOWS, los requisitos navegacionales de una aplicación web se obtienen añadiendo una "vista navegacional" (*mapa navegacional*) sobre el Modelo de Objetos de OO-Method, indicando el conjunto posible de caminos navegacionales que se le proporcionarán al usuario. Al definir este modelo dependiente del modelo de objetos, se crea una relación fuerte de dependencia entre ambos modelos, integrándolos de una manera clara y posibilitando la comprobación automática de propiedades semánticas del sistema. La semántica navegacional de las aplicaciones hipermediales se captura en función de cada agente del sistema identificado en el modelo de objetos, adaptando o personalizando el acceso en función de las necesidades de cada tipo de usuario. Estos mapas son descritos usando una notación intuitiva basada en UML [15].

El modelo de navegación está compuesto por un conjunto de **mapas de navegación** (uno por cada agente) que representan y estructuran la visión global del sistema para cada tipo de usuario, definiendo su navegación permitida. Éste se representa directamente usando un grafo dirigido en el cual los nodos son los *contextos navegacionales* y los arcos son los *enlaces (o vínculos) de navegación*. En este nivel de abstracción, sólo nos interesa especificar qué contextos conformarán nuestro mapa de navegación y desde dónde serán alcanzables. Existen dos posibilidades: que los nodos (contextos) navegacionales sean alcanzables desde cualquier ubicación en el sistema (llamados *contextos de exploración, E*) o que los nodos sólo sean alcanzables siguiendo un camino predeterminado de pasos de navegación (llamados *contextos de secuencia, S*). Los contextos de exploración definen unos enlaces de navegación implícitos (enlaces de exploración, representados por una flecha con líneas discontinuas) que surgen del agente y terminan en el contexto, indicando que ese agente siempre puede activar ese enlace y alcanzar el contexto de exploración. Además, se puede marcar uno de estos enlaces de exploración como *default o home* (aparece una "*H*" asociada al enlace), indicando que cuando el usuario se conecte, navegará automáticamente a este contexto. En la Figura 1 se puede apreciar un mapa de navegación genérico para el agente *Agente* donde aparecen dos contextos de exploración (marcados con la etiqueta *E*) y uno de secuencia (marcado con la etiqueta *S*).

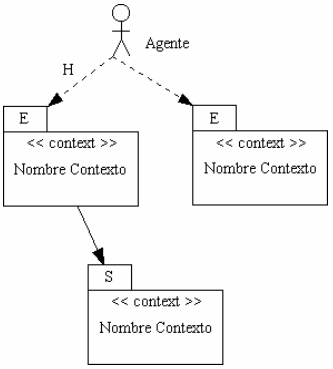

Fig. 1. Mapa de Navegación

Un **contexto navegacional** es una *Unidad de Interacción Abstracta* que representa una vista sobre un conjunto de datos y/o servicios (del esquema conceptual) accesible para un usuario en un determinado momento. Es una *Unidad* porque constituye el elemento lógico básico de creación de la navegación permitida en los mapas navegacionales. De *Interacción* porque representa una interacción con el usuario (espera una acción/respuesta por parte del usuario, bien de navegación, bien de activación de un servicio), y *Abstracta* porque sólo se especifica qué datos y/o servicios se visualizarán en el contexto, pero no cómo se presentarán. Gráficamente se representa como un paquete UML estereotipado con la palabra «*context*». Está compuesto por un conjunto de **clases navegacionales**, estereotipadas con la palabra reservada «*view*» (Figura 2), que hacen referencia a clases identificadas en el modelo de objetos. Con ellas se puede definir la visibilidad (vista) ofertada al agente en este nodo, tanto de los atributos de la clase como de los servicios que puede activar. Este conjunto de atributos y servicios debe ser un subconjunto válido de atributos y servicios con los que el agente tenga definida una relación de agente en el modelo de objetos (relación de visibilidad de atributos y/o servicios en OO-Method).

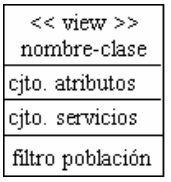

Fig. 2. Clase Navegacional

Asociado a los servicios pueden aparecer **enlaces de servicio**. Estos enlaces indican el contexto destino que se alcanzará después de la ejecución del servicio. Además, se puede especificar un **filtro de población**, que mediante una fórmula bien formada

en función de atributos de la clase navegacional sobre la que se define, indican un filtrado de objetos que se llevará a cabo.

En un contexto navegacional debe aparecer una clase navegacional principal, llamada **clase directora** y opcionalmente otras que contribuyen en complementar la información de esta clase, llamadas **clases de complementarias**. Las clases navegacionales están unidas entre sí por relaciones binarias unidireccionales que pueden ser definidas sobre una relación de agregación o de herencia existente entre las dos clases en el modelo de objetos[1]. En el caso de que exista más de una relación definida en el modelo de objetos entre estas dos clases, se deberá especificar una propiedad adicional de la relación que indique que rol se está utilizando, para eliminar la ambigüedad. Se pueden definir dos tipos de relaciones entre clases navegacionales: una **relación de dependencia de contexto** (se representa gráficamente mediante flechas discontinuas) que indica una recuperación de información relacionada de las instancias de la clase complementaria a través de la relación de agregación o herencia sobre la que está definida la relación, y una **relación de contexto** (gráficamente, flechas continuas), que es una relación de dependencia de contexto que además define una navegación a un nodo navegacional destino, causando la aparición de un vínculo de navegación en el mapa navegacional asociado. Las relaciones de contexto tienen las siguientes propiedades:

- *atributo de contexto*, que indica el contexto destino de la navegación
- *atributo de enlace*, que especifica qué atributo (normalmente de la clase navegacional final de la relación) se utilizará como *ancla* para activar la navegación al contexto destino
- *atributo de rol*, que indica el rol de la relación de agregación o herencia que estamos utilizando. Se utiliza para eliminar la ambigüedad en caso de existencia de más de una relación entre las dos clases. Es opcional si entre las dos clases existe una única relación definida en el modelo de objetos

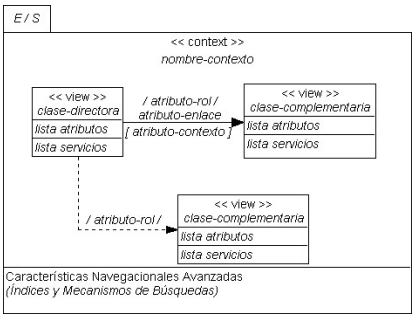

Fig. 3. Contexto de Navegación

Usando las primitivas descritas anteriormente es posible especificar completamente la semántica asociada a los requisitos de navegación para aplicaciones web. Sin embargo, y debido al carácter de Internet, nos hará falta especificar algunas propiedades adicionales de diseño para facilitar el acceso a la información deseada y que pueden

[1] No puede existir ninguna clase navegacional en un contexto que esté inconexa, es decir, que no esté relacionada con ninguna otra clase navegacional.

llegar a ser cruciales para el sistema. Estas propiedades tienen que ver con la incorporación de *estructuras de acceso* y la definición de *mecanismos de búsqueda* de la población dentro de un contexto. Estas características se abordarán en el siguiente apartado.

Diseño Navegacional Avanzado.

Una vez que se han construido los mapas navegacionales, la semántica navegacional del sistema ya ha sido capturada. Sin embargo, se pueden definir mecanismos adicionales que estructuren el acceso y permitan realizar búsquedas de información dentro de un nodo navegacional. Ambos mecanismos permitirán explorar y facilitar el acceso a la misma información, sin implicar navegación. Estas características son recogidas en la zona de características navegacionales avanzadas de un contexto de navegación (ver Figura 3).

Un **índice** proporciona un acceso *indexado* (por alguna propiedad propia o de un objeto relacionado) a los objetos principales del contexto (objetos de la clase directora). Por ejemplo, en un Museo Virtual podría existir un contexto que presentara las pinturas expuestas. En este contexto se podría definir un índice de acceso a las pinturas según corrientes pictóricas. Al seleccionar un elemento de este índice, se recuperarían todas las pinturas de esta corriente, dentro del mismo contexto.

Existen dos tipos de índices que pueden ser especificados (ver Figura 4):

- *índices de atributos (ATTRIBUTES-INDEX)*, que se definen sobre uno o varios atributos de la clase directora. Al menos uno de estos atributos actuará como *atributo de enlace* (atributo/s que servirá/n para activar el índice). Se creará un índice donde aparecerán sólo los valores de los atributos especificados de la población de objetos del contexto. Se puede indicar que no aparezcan duplicados (*DISTINCT VALUES*), para que sólo se recuperen los valores distintos.
- *índices de relación (RELATION-INDEX)*, que se definen sobre uno o varios atributos de una clase relacionada en el modelo de objetos con la clase directora del contexto. Al menos uno de estos atributos servirá como *atributo de enlace* (atributo/s que servirá/n para activar el índice). Si existe más de una relación entre la clase directora y la clase del índice en el modelo de objetos, será necesario especificar un *atributo de rol* para eliminar la ambigüedad. Se creará un índice con todos los valores de atributos de los objetos de la clase relacionada. Se puede indicar que no aparezcan duplicados (*DISTINCT VALUES*), para que sólo se recuperen los valores distintos.

ATTRIBUTES INDEX *nombre*
ATTRIBUTES *lista-atributos*
LINK-ATTRIBUTES *lista-atributos*
[DISTINCT VALUES]

RELATION INDEX *nombre*
ON RELATION *rol-relación*
ATTRIBUTES *lista-atributos*
LINK-ATTRIBUTES *lista-atributos*
[DISTINCT VALUES]

Fig. 4. Plantillas de definición de *índices* asociados a contextos

Adicionalmente, sobre un contexto se pueden especificar **mecanismos de búsqueda**, expresados como filtros de información. Estos filtros permiten restringir el espacio de objetos de la clase directora recuperados en función de una expresión-condición basada sobre alguna propiedad (atributo) de la clase directora u obtenida

por una relación entre clases. Esta expresión se puede especificar en tiempo de modelado o la puede introducir el usuario en tiempo de ejecución. La Figura 5 muestra la plantilla de definición de un filtro. Existen tres tipos:

- *exacto*, que toma un único valor y devuelve el conjunto de instancias de la clase directora cuyo valor de atributo coincida exactamente con el valor indicado por el usuario
- *aproximado*, que toma un único valor y devuelve el conjunto de instancias de la clase directora cuyo valor de atributo sea semejante al valor indicado por el usuario
- *rango*, que toma dos valores, un máximo y un mínimo y devuelve conjunto de instancias de la clase directora cuyo valor de atributo esté entre estos valores[2]. Si sólo se especifica uno de estos valores, se acota únicamente por un extremo

FILTER *nombre*
ATTRIBUTE *atributo*
TYPE { *exact* | *like* | *range* }
{ EXPRESSION *condición* }

Fig. 5. Plantilla de definición de un *filtro* de contexto

La Figura 6 muestra un ejemplo de un filtro completamente especificado en tiempo de modelado que recupera los libros *best-sellers*. Se puede considerar que un libro es un best-seller cuando se han vendido más de 1.000.000 de copias. Al seleccionar este filtro, la población de libros se filtraría por aquellos libros que cumplen esta condición.

FILTER libros_best-sellers
ATTRIBUTE vendidos
TYPE range
EXPRESSION vendidos > 1.000.000

Fig. 6. Filtro especificado completamente en tiempo de modelado

Otras primitivas que se pueden especificar en el modelo de navegación OOWS:

- *Sesión*. Primitiva de modelado que nos permite realizar cierto control cuando un usuario se conecta al sistema. Existen básicamente dos momentos donde se pueden lanzar operaciones: ejecutar un conjunto de servicios cuando el usuario inicia o cuando termina la sesión de interacción con el sistema. El agente de estos servicios es siempre el propio usuario. La sintaxis es:

#Clase.Servicio1#	##Clase.Servicio1##
#Clase.Servicio2#	##Clase.Servicio2##
...	...
Eventos de inicio de sesión	*Eventos de fin de sesión*

- *Agente conectado*. Esta primitiva proporciona capacidad expresiva para acceder al usuario conectado a la aplicación en tiempo de ejecución. Esta expresividad es la base para personalizar el acceso y la recuperación de información según el usuario

[2] Este atributo solo tiene sentido en atributos de tipo numérico o de texto.

concreto. Se puede utilizar en fórmulas de filtro, en condiciones de navegación, etc. Se representa mediante el término *#Clase_Agente#*.

– *Condición de navegación*. La navegación capturada mediante las relaciones de contexto se caracteriza por ser estática, es decir, enlaza unívocamente dos contextos navegacionales. Sin embargo, dada la naturaleza de las aplicaciones web se hacen necesarios mecanismos que expresen dinámicamente condiciones de navegación que sean evaluadas en tiempo de ejecución. Éstas condiciones permiten especificar restricciones que deben satisfacerse para que se pueda producir una navegación. Se expresan en las relaciones de contexto, mediante el atributo de contexto, explicitando la condición o condiciones que deben cumplirse para alcanzar el contexto destino.

Mediante el uso de estas características avanzadas, se pueden complementar algunos requisitos navegacionales adicionales del sistema. A continuación vamos a proponer un modelo que capture los requisitos de presentación de información del sistema.

Modelo de Presentación

Una vez definido el modelo de navegación que captura la semántica navegacional del sistema, debemos asociar características de presentación al sistema. Para ello se introduce un nuevo modelo, el Modelo de Presentación, que complementa la información capturada en el modelo de navegación para la creación de interfaces con información de presentación. En este modelo se utilizan los nodos o contextos navegacionales como entidades básicas donde se definen estas propiedades de presentación adicionales.

La manera de especificar los requisitos de presentación se basará en el uso de unos patrones de presentación simples que se podrán asociar a los distintos elementos que forman un nodo de navegación. Existirán propiedades de presentación que podrán ser aplicadas a nivel de un contexto de navegación, a nivel de estructuras de acceso y/o mecanismos de búsquedas (filtros), y a relaciones (navegacionales) entre clases.

Los patrones de presentación de información que se pueden especificar son:

– *Paginación de información*. Este patrón permite capturar la semántica *scrolling* de información. Cuando se especifica paginación, el conjunto de instancias que deban ser presentadas serán "troceadas" en "bloques lógicos", de manera que en una misma "pantalla" sólo aparezcan unas cuantas instancias del conjunto de todas las posibles. Se proporcionarán mecanismos para avanzar y retroceder entre las distintas páginas lógicas que se obtienen, pudiendo especificar el tipo de paginación como *secuencial* (se proporciona acceso al siguiente, al anterior, al primero y al último bloque), o *aleatorio* (cuando además de los anteriores, el usuario puede acceder directamente a un bloque intermedio). Se podrá indicar además una *cardinalidad* en la paginación, que indicará el número de instancias que se recuperarán. Puede ser de dos tipos: *estática*, cuando se define en tiempo de modelado y *dinámica*, cuando se permite modificar en tiempo de ejecución por el usuario. En este último caso se indicará un número de instancias a mostrar por defecto (que podrá ser modificable) o una selección de valores posibles entre los que el usuario deberá elegir. Otra propiedad de la paginación es la *circularidad*. Si se pagina con esta propiedad

activada, la siguiente instancia de la última instancia será la primera instancia, y la instancia anterior a la primera instancia será la última instancia. Cuando se define que un contexto está paginado, las instancias sobre las que se pagina son las de la clase directora. Si se aplica a una estructura de acceso índice o un filtro de búsqueda, se pagina el conjunto de resultados obtenidos. Por último, se puede indicar paginación sobre una relación navegacional (relación de dependencia de contexto y relación de contexto) entre dos clases cuando la cardinalidad en el modelo de objetos de la clase complementaria destino sea "muchos". En este caso se crea una paginación que se aplicará únicamente sobre el conjunto de objetos relacionados que se presentarán.

- *Ordenación*. Este patrón permite definir una ordenación de la población de una clase según el valor de uno o más atributos sobre los que se aplica. La ordenación puede ser: *ascendente (asc)* o *descendente (desc)*. En el caso de especificar varios atributos, cada atributo tendrá un carácter de ordenación ascendente o descendente, y ésta se aplicará jerárquicamente, empezando por el primer atributo, y siguiendo por los demás sucesivamente. Este patrón se puede aplicar a clases navegacionales, indicando cómo se van a ordenar las instancias de la clase (los atributos sobre los que se ordene deben aparecer especificados en el contexto); análogamente al caso anterior, se pueden aplicar a estructuras de índices y a filtros de búsqueda, ordenando los resultados obtenidos por alguno/s de los atributo/s especificado/s por estos mecanismos.
- *Patrón de presentación*. Existen cuatro modos: *registro, tabular, maestro-detalle* (pudiendo indicar en éste último caso como presentar el elemento detalle, recursivamente) y *árbol*. Se pueden aplicar a:
  - *relaciones de navegación* (relación de dependencia de contexto y relación de contexto). El patrón de presentación definirá el modo en que la información de las instancias relacionadas será presentada. Los patrones *registro* y *tabular* son indicados para relaciones "1 a 1", mientras que los dos últimos son adecuados para relaciones "1 a muchos" ó "muchos a muchos". El patrón *árbol* es también muy adecuado para representar relaciones reflexivas.
  - la *clase directora*, indicando el modo en que se verán las instancias de esta clase. El patrón *maestro-detalle* no se puede aplicar directamente a esta clase.

Con estos patrones sencillos de presentación de información, combinados con la información de navegación definida en el modelo de navegación, podemos capturar los requisitos básicos para la construcción de interfaces del sistema, a nivel de modelado conceptual. Este repositorio de información será utilizado por el generador (compilador) para generar las distintas interfaces para cada usuario, dentro de la arquitectura de aplicación web que propone el método OOWS.

En el siguiente apartado se va a presentar un caso de estudio donde se ha aplicado el método propuesto, siguiendo el proceso definido de construcción de aplicaciones web. Al final se presenta un ejemplo de interfaz generada teniendo en cuenta las primitivas especificadas en los modelos para este caso de estudio.

Un caso de estudio: La tienda virtual de venta de discos *DiscoWeb*

En esta sección se aplica la aproximación OOWS en el desarrollo de una aplicación web para la compra discográfica por Internet. Los requisitos del sistema se presentan a continuación:

"El sistema está orientado a la venta *on-line* de álbumes de música, organizados por categorías. Existen dos tipos de usuarios de esta aplicación: los Administradores y los Usuarios Navegantes. Los primeros son los encargados de la gestión básica de los álbumes que se ponen en venta. Los segundos son los usuarios comunes (compradores) de esta aplicación. La funcionalidad del sistema es:

- (*Usuarios Navegantes*). Las compras que se realicen se deberán ir incluyendo, simbólicamente, en una cesta de la compra; el usuario podrá consultar en cualquier momento el contenido de su cesta y realizar modificaciones sobre su contenido. Esta cesta de la compra se creará en el momento en el que se reciba la petición de entrada en el sistema y pertenecerá al usuario que está navegando en ese momento; todas las operaciones que el usuario realice sobre el sistema se harán de forma anónima, de modo que el usuario no deberá identificarse (registrarse) hasta que no vaya a confirmar su compra; para comprar un álbum se deberá llegar a él a través del autor o de la categoría a la que pertenece; desde la página de inicio podremos acceder a un listado de autores o a un listado de categorías y desde ahí al listado de los álbumes del autor o de la categoría que hayamos seleccionado; cuando seleccionemos un álbum de la lista, se mostrarán todos los datos de ese álbum y se podrá comprar. Esto hará que el álbum sea incluido en la cesta de la compra de ese usuario y que se muestre su contenido actual; mientras veamos el contenido de la cesta, podremos cambiar el número de unidades que se desea adquirir de cada álbum de los comprados hasta el momento o eliminar alguna de las compras de la cesta; cuando se decida confirmar la compra se realizarán dos acciones: la primera consistirá en crear una factura (para lo que el comprador debe haberse identificado) y la segunda será reducir el stock de los álbumes comprados; cuando se haya confirmado una compra, ya no se podrá modificar el contenido de la cesta.
- (*Administradores*). Gestionar y mantener el catálogo de productos del sistema, así como mantener la lista de autores que poseen discos en la tienda".

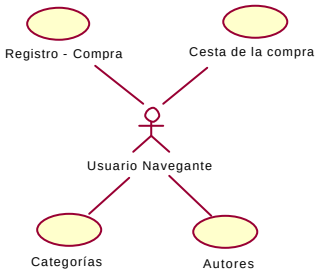

Fig. 7. Caso de uso del agente *Usuario Navegante*

El caso de estudio empieza con la elicitación de los requisitos del sistema en la fase de **Especificación del Problema.** El sistema se divide en la funcionalidad relevante y estos requisitos son descritos mediante un diagrama de casos de uso. En la Figura 7 se muestra el diagrama de caso de uso para el agente Usuario Navegante. Del mismo modo se han descrito los requisitos para el agente Administrador (no aparecen en este artículo por falta de espacio y por carecer de aspectos interesantes). En la fa-

se de modelado conceptual se construyen los siguientes modelos: Modelo de Objetos, Modelo Dinámico, Modelo Funcional, Modelo Navegacional y Modelo de Presentación. Estos dos últimos recogen las extensiones incorporadas a OO-Method.

La Figura 8 muestra el **Modelo de Objetos** del caso de estudio. La *cesta* de la compra participa directamente en el proceso de compra *on-line* del internauta *Usuario Navegante*. Para su construcción se han estudiado los requisitos funcionales del sistema desde un punto de vista OO, y se ha asociado esta funcionalidad a cada usuario, según los casos de uso planteados (esta información no está visible por motivos de espacio del artículo).

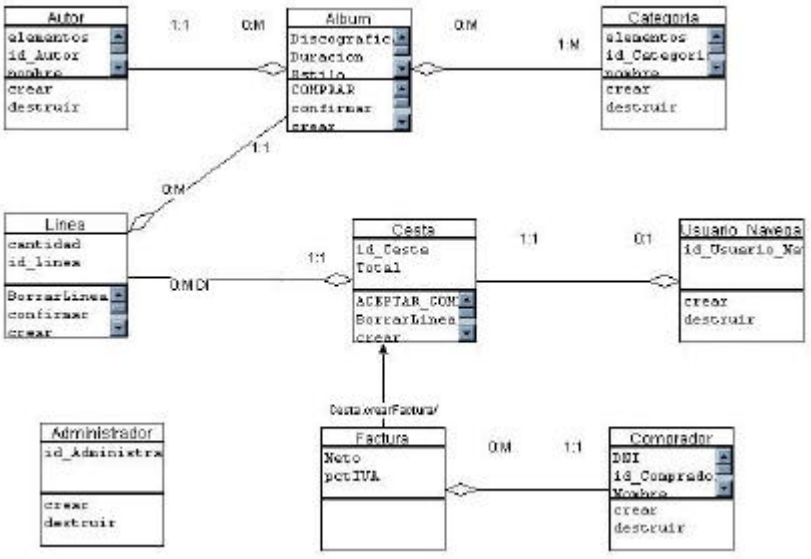

Fig. 8. Modelo de Objetos de DiscoWeb

Después del Modelo de Objetos, se construye el **Modelo Dinámico** donde se describen las vidas válidas de los objetos representando el comportamiento de cada clase del sistema según la interpretación descrita por OO-Method. La Figura 9 presenta una porción del Modelo Dinámico de la clase *Cesta*:

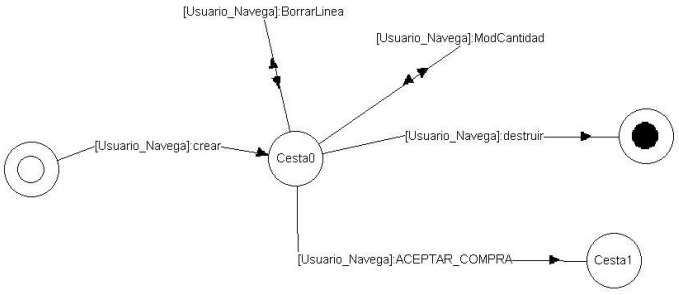

Fig. 9. Porción del Modelo Dinámico de la clase *Cesta*

El **Modelo Funcional** captura la semántica asociada a los cambios de estado de los objetos. El valor de cada atributo es modificado dependiendo de la acción que activó el cambio de estado, de los argumentos de dicho evento y del estado actual del objeto. La Figura 10 muestra un ejemplo para la clase *Linea*, donde se halla definida la siguiente evaluación:

```
Atributo: Cantidad
Categoría: De Estado     Evento: ModCantidad(p_nuevaCantidad)
Efecto: = p_nuevaCantidad      Condición: p_nuevaCAntidad > 0
```

Fig. 10. Parte del Modelo Funcional de la clase *Linea*

A continuación se construye el **Modelo de Navegación** donde se estructurará el acceso de cada usuario al sistema. La Figura 11 presenta el mapa de navegación del agente *Usuario Navegante* con sus contextos de navegación que han sido identificados en las primeras fases de especificación del sistema. También aparecen sobre el mapa los servicios que son ejecutados al iniciar y finalizar una sesión. Cuando el servidor Web recibe una petición de un internauta, ejecuta el servicio *crear* del *Usuario Navegante* asociándole además una *Cesta*. Cuando el *Usuario Navegante* abandona el sistema se ejecuta el servicio *destruir*, eliminando además, si no ha sido confirmada, su *Cesta* asociada.

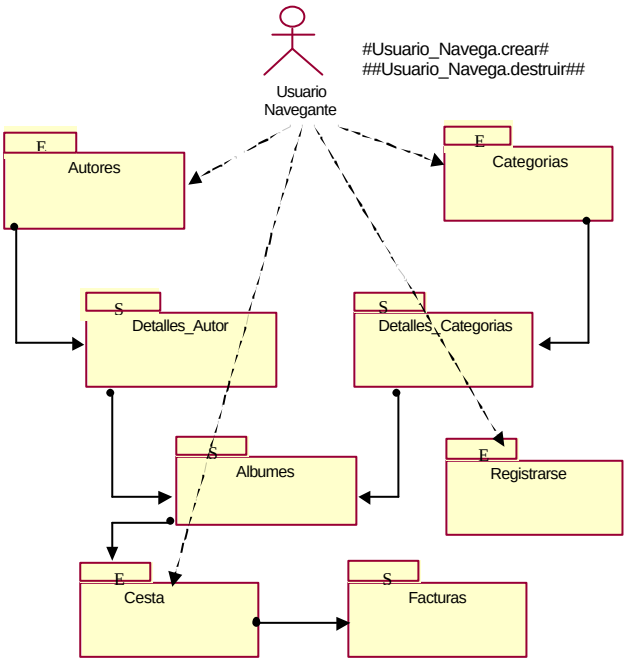

Fig. 11. Mapa de Navegación del agente *Usuario Navegante*

En este mapa de navegación se aprecia que el *Usuario Navegante* siempre tendrá disponibles los contextos (marcados como contextos de exploración) *Autores, Categorías, Cesta* y *Registrarse*. A partir de estos, y siguiendo diferentes caminos navegacionales, podrá alcanzar los demás (*Detalles_Autor, Detalles_Categoría, Albumes y Facturas*).

En la Figura 12 se describe con detalle el contexto *Autores*, donde se recupera la información sobre un autor (su nombre), los álbumes que están disponibles de este autor (título, año y precio) y el nombre de la categoría del álbum. Seleccionando el título de un álbum podremos navegar al contexto *Álbumes*, donde se proporcionará información adicional del álbum y podremos comprarlo. Además, se ha definido una estructura de acceso índice de tipo atributo, que permitirá acceder a los autores por su *letra_inicial* (atributo derivado definido en la clase *Autor*). También se ha definido un filtro de tipo aproximado para facilitar la búsqueda de autores por su nombre.

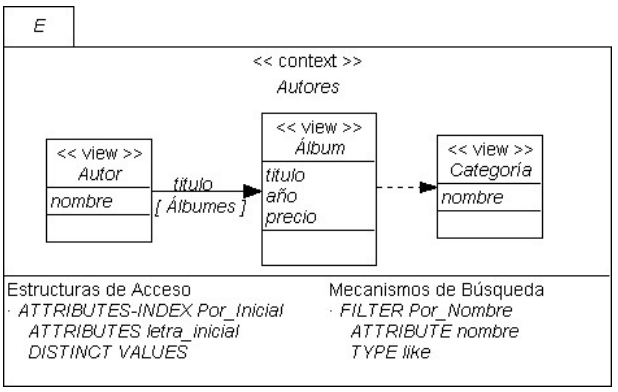

Fig. 12. Contexto *Autores* para el agente *Usuario Navegante*

Finalmente, se construye el **Modelo de Presentación** donde se captan los requisitos de presentación de información para cada contexto del mapa de navegación. En la Figura 13 se muestra como ejemplo la plantilla de presentación asociada al contexto *Autores*. En ella se especifica que los objetos de la clase directora se presentarán en modo *tabular* y el contexto (objetos de la clase directora) estará paginado con una cardinalidad estática de 1 elemento, con posibilidad de acceso *secuencial* y *circular*. El patrón de presentación asociado a la relación de contexto definida entre un *Autor* y sus *Album*es será *maestro-detalle*, con el detalle en modo *tabular* y con una paginación de cardinalidad estática de 5 elementos, con acceso *secuencial*, *circular*. Se ha definido una ordenación de los elementos de la clase *Album* por el *año* de modo *ascendente* y la relación de contexto definida entre la clase *Album* con la clase *Categoría* se presentará en modo *tabular* (relación "1 a 1").

Después de construir estos modelos, el proceso llega a su segunda fase, **Desarrollo de la Solución**, donde en una estrategia de compilación de modelos, se obtiene el prototipo software completo de manera automática, siguiendo la especificación realizada del sistema. En la Figura 14 aparece una posible interfaz de usuario que representa correctamente los requisitos, tanto navegacionales como de interfaz, especificados para el agente *Usuario Navegante*.

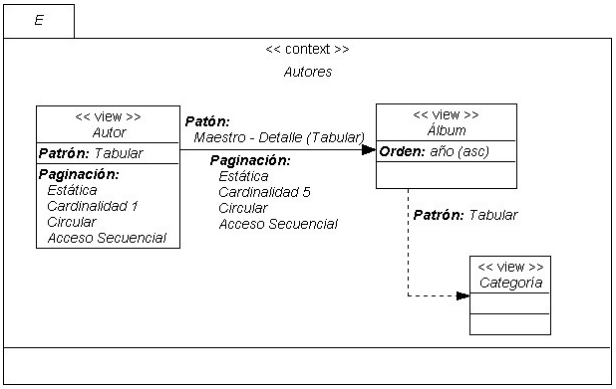

Fig. 13. Información adicional de presentación del contexto *Autores*

Fig. 14. Página generada a partir del contexto navegacional *Autores*

Como se puede apreciar en este contexto generado (Figura 14), esta implementación recoge las características navegacionales y de presentación de información que se habían especificado en la definición del contexto (Figuras 12 y 13). Se puede apreciar que existe un enlace desde esta página (en realidad desde cualquier página, ya que se el *frame* izquierdo es común a todas las páginas) a cada uno de los contextos de exploración especificados (*Autores*, *Categorías*, *Cesta* y *Registrarse*). Esta página proporciona información sobre un autor y sus álbumes disponibles, mostrando su título, año, precio y categoría. Además, aparece el índice que se había especificado, usando el atributo *letra_inicial* de la clase *Autor* y también el mecanismo de búsqueda de

de autores por su atributo nombre. Si nos fijamos en aspectos de presentación de información, la paginación del contexto (objetos de la clase directora *Autor*) se realiza elemento en elemento (cardinalidad 1). Es por esto que sólo nos aparece información sobre un autor o grupo (*Queen*) y se nos proporcionan mecanismos para avanzar o retroceder secuencialmente. La información complementaria sobre los álbumes para este grupo aparece en modo *maestro-detalle* (con el detalle en modo *tabular*) y además paginado de 5 en 5 elementos, como había sido explicitado.

Conclusiones

En este artículo hemos presentado una propuesta metodológica para la construcción de aplicaciones web dentro un marco de trabajo OO en ambientes de compilación automática de modelos. Este marco definido integra formalmente diferentes modelos para dar un soporte completo a la especificación de este tipo de sistemas software, además de definir un proceso metodológico guiado. El trabajo realizado ha consistido en extender un método formal existente OO (OO-Mehod), dotándolo de la expresividad necesaria para construir soluciones web. Esta expresividad ha sido capturada en: un *Modelo de Navegación* que captura la estructura navegacional asociada a cada usuario del sistema, proporcionando el acceso a la información, y a la funcionalidad, permitiendo definir estructuras de acceso y mecanismos de búsqueda que faciliten el acceso a la información deseada; un *Modelo de Presentación*, que refleja las características esenciales de presentación de información de los nodos navegacionales mediante (unos pocos) patrones básicos de presentación.

Se ha aplicado el método a un caso de estudio, siguiendo el proceso propuesto de construcción de un sistema web. Adicionalmente, se han realizado trabajos orientados a dar soporte a:

- la introducción de características de personalización a nivel de modelado conceptual, para crear sistemas adaptativos
- la definición de operadores conceptuales orientados al reuso conceptual
- la construcción de interfaces web declarativas, basadas en XML

Actualmente estamos aplicando el método a numerosos casos de estudio reales, extendiéndolo incorporando nuevas primitivas navegacionales detectadas, y definiendo sus patrones de traducción software a plataformas destino.

En este ámbito se está se rediseñando la estrategia de generación de código basada en componentes para adaptarla a una arquitectura software destino y completamente distribuida, basada y orientada en servicios web.

Referencias

[1] Abrahao S., Fons J., and Pastor O. *Conceptual Modeling of Personalized Web Applications*. Springer–Verlag (LNCS). Editores: Paul De Bra, Peter Brusilovsky, Ricardo Conejo. 2nd Intenational Conference on Adaptive Hypermedia and Adaptive Web Based Systems. Málaga, España, Junio, 2002

[2] Amazon.com Tienda Virtual de Comercio Electrónico. http://www.amazon.com
[3] Baresi L., Garzotto F., Paolini P. *From Web Sites to Web Applications: New Issues for Conceptual Modeling*. ER'2000 Workshop on Conceptual Modeling and the Web, LNCS 1921. Springer-Verlag, 2000
[4] Baumeister H., Koch N., and Mandel L. *Towards a UML extension for Hypermedia Design.* In UML'99 The Unified Modeling Language, Springer Verlag. LNCS 1723. Fort Collins, USA, Octubre 1999.
[5] Cachero C., Garrigós I., and Gómez J. *Personalización de Aplicaciones en OO-H*. In Proc. V Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software, IDEAS'02. ISBN: 959-7164-08-6. La Habana, Cuba, Abril, 2002
[6] Ceri S., Fraternali P., Bongio A. *Web Modeling Language (WebML): a Modeling Language for Designing Web Sites*. In WWW9, Vol. 33 (1-6), pp 137-157. Computer Networks, 2000
[7] De Troyer O. and Leune C. *WSDM: A user-centered design method for Web sites*. In Proc. of the 7th International World Wide Web Conference, 1997
[8] Fons J., Valderas P., and Pastor O. *Specialization in Navigational Models*. Anales JAIO, 31st. Argentine Conference on Computer Science and Operacional Research. Iberoamerican Conference on Web Engineering 2002 (ICWE'02), Santa Fe, Argentina, Septiembre, 2002
[9] Garzotto F., Paolini P. and Schwabe D. *HDM - A Model-Based Approach to Hypertext Application Design*. ACM Transactions on Information Systems, vol. 11 (1), pp. 1-26. 1993
[10] Gómez J., Cachero C., and Pastor O. *Extending a Conceptual Modeling Approach to Web Application Design*. Proc. Conference on Advanced Information Systems Engineering (CAiSE'00), Springer- Verlag, LNCS 1789, pp. 79-93, 2000
[11] Isakowitz T., Stohr E. A., and Balasubramanian V. *RMM: A Methodology for Structured Hypermedia Design*. CACM: Communications of the ACM., pages 34-44, Agosto, 1995
[12] Jacobson I., Christerson M., Jonsson P., Overgaard G. *OO Software Engineering , a Use Case Driven Approach*. Reading, Massachusetts. Addison-Wesley, 1992
[13] Lange D. *An Object-Oriented Design Method for Hypermedia Information Systems*. Proc. Hawaii International Conference on System Science, 1994
[14] Mecca G., Merialdo P., Atzeni P. and Crescenzi V. *The Araneus Guide to Web-Site Development*. Technical Report, University of Roma. 1999
[15] Object Management Group. *Unified Modeling Language Specification Version 1.4 draft*. Technical report, www.omg.org, February 2001
[16] Paolini P. and Fraternali P. *A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications*. In Proc. EDBT98 Conference, Valencia, España, Marzo, 1998
[17] Pastor O., Insfrán E., et. al. *OO-Method: An OO Software Production Environment Combining Conventional and Formal Methods*. Proc. Conference on Advanced Information Systems Engineering (CAiSE), LNCS 1250, Springer- Verlag, pp. 145-159, 1997
[18] Pastor O., Abrahao S., and Fons J. J. *An Object-Oriented approach to automate web applications development*. In 2nd International Conference on Electronic Commerce and Web Technologies (EC-Web'01), Springer-Verlag, LNCS 2115. ISBN: 3-540-42517-9. Munich, Germany, Septiembre, 2001
[19] Pastor O., Pelechano V., Insfrán E,. and Gómez J. *From Object Oriented Conceptual Modeling to Automated Programming in Java*. 17th International Conference on Conceptual Modeling (ER'98). Springer-Verlag, LNCS 1507, pp. 183-196. Singapore, November, 1998
[20] Pastor O., Abrahao S., and Fons J. *Building E-Commerce Applications from Object-Oriented Conceptual Models*. Newsletter of the ACM SIGecom Exchanges, volume 2.2, ACM Press, pp. 24-32. June, 2001
[21] Rossi G., Schwabe D., and Guimaraes R. *Designing personalized web applications*. In Proc. of the WWW10, Hong Kong, May 2001
[22] Schwabe D., Rossi G. and Barbosa D.J. *Systematic Hypermedia Application Design with OOHDM*. Proc. ACM Conference on Hypertext. pp.166. 1996