OO-$\mathcal{H}$Method: un método de diseño de lugares Web

Cristina Cachero[1], Jaime Gómez[1], Oscar Pastor[2]

[1] DLSI. Universidad de Alicante
e-mail:{ccachero,jgomez}@dlsi.ua.es
[2] DSIC. Universidad Politécnica de Valencia
e-mail:opastor@dsic.upv.es

Resumen Este artículo introduce OO-$\mathcal{H}$Method, una extensión del método OO-Method que permite el desarrollo de aplicaciones en ambientes Web. OO-$\mathcal{H}$Method hereda parte de sus conceptos de otros modelos y metodologías existentes, especialmente OOHDM y HDM-lite, para a continuación modificar y ampliar su semántica de modo que sea posible modelar la funcionalidad específica de interfaces de aplicación. Para el modelado de estas interfaces se introducen los diagramas de acceso navegacional (DAN). Los DAN se generan mediante la captura de los Requerimientos Navegacionales de cada tipo de usuario sobre la vista del diagrama de clases OO-Method relevante para ese tipo de usuario. Todos los diagramas son almacenados en un repositorio que permite la posterior generación automática de la interfaz Web que dialogue con los módulos de aplicación generados en OO-Method utilizando la tecnología elegida por el diseñador.

1 Introducción

Los numerosos estudios y avances realizados por la comunidad científica en el campo de tecnologías Web, así como su aplicación a otras áreas de conocimiento como la Ingeniería del Software, han propiciado la aparición de aplicaciones hipermediales cada vez más complejas y dinámicas en términos de estructura, contenido, comportamiento e interfaz. Según aumenta esta complejidad, procesos de desarrollo 'creativos' o poco estructurados causan una serie de problemas ampliamente documentados [9,5,1,7,17,14,12,10] tanto de cara al desarrollador como de cara al usuario final.

Es por ello por lo que en los últimos años se ha investigado de manera exhaustiva, por un lado, sobre las características deseables de las aplicaciones hipermediales, y por otro sobre los conceptos y procesos que intervienen en el desarrollo efectivo de estas aplicaciones. También se han propuesto diversos métodos de modelado de aplicaciones Web genéricas, entre los cuales se pueden destacar los desarrollados en los proyectos Araneus [13,1], Autoweb [7] o Strudel [6], que además constituyen entornos de desarrollo completos Web-Database [4].

En este artículo se presenta una extensión de OO-Method, cuya especificación detallada se puede encontrar en [15,16]. Esta extensión, conocida como OO-$\mathcal{H}$Method, donde la $\mathcal{H}$ expresa su adecuación para la generación de Interfaces Hipermediales, permite capturar la expresividad necesaria para confeccionar interfaces de usuario de aplicaciones Web compatibles con modelos OO-Method ya generados. Un nuevo conjunto de diagramas (definidos

* Este artículo ha sido escrito con el patrocinio de la Conselleria de Cultura, Educació y Ciència de la Generalitat Valenciana

tanto en el ámbito del modelado conceptual como en el de ejecución), para los que se detallan características semánticas y de notación, permiten cubrir los aspectos observacional, navegacional y de presentación, que han demostrado ser claves en el diseño de aplicaciones para la Web. Siguiendo la filosofía de OO-Method, OO-$\mathcal{H}$Method define un catálogo finito de Patrones de Interfaz. La conversión de estos patrones a su correspondiente representación software permite la generación automática del esqueleto de la aplicación en arquitecturas Web. El nivel de presentación se completa con la definición de las características visuales mediante la utilización de plantillas [6].

El resto del artículo está estructurado de la siguiente forma: en la sección 2 se introduce una descripción general de los dos modelos (Conceptual y de Ejecución) de OO-$\mathcal{H}$Method, así como del diagrama de acceso navegacional (DAN) que acompaña al Modelo Conceptual. En la sección 3 se detallan las conclusiones y en la sección 4 los trabajos futuros.

2 OO-$\mathcal{H}$Method

OO-$\mathcal{H}$Method es un modelo de estructuración semántica de interfaces Web, y como tal se centra en actividades globales (authoring in the large) [9], es decir, en clases y estructuras, dejando de lado el contenido de los nodos de información (authoring in the small). Este modelo captura las primitivas y conceptos considerados clave en el ámbito de sistemas hipermediales y aplicaciones Web. Estos conceptos aparecen de manera recurrente en los distintos modelos estudiados. Para nuestro modelo han resultado especialmente relevantes HDM-lite [7] y OO-HDM [19], con quien comparte además el enfoque OO.

OO-$\mathcal{H}$Method se integra dentro OO-Method, y extiende el conjunto de elementos de notación gráfica para permitir la captación de las propiedades inherentes a las Interfaces de Usuario y construir así un Modelo Conceptual de Interfaz, es decir, un modelo abstracto de interacción entre usuario y aplicación. Posteriormente, un Modelo de Ejecución de Interfaz determinará la forma de implementar el Modelo Conceptual en un entorno de desarrollo concreto.

El Modelo Conceptual se encarga de la abstracción de los principales rasgos de la Interfaz de Usuario, sin preocuparse de aspectos de implementación. Define por tanto qué información puede ver cada tipo de usuario (agente) y qué caminos de navegación existen para cubrir sus requerimientos navegacionales. A partir de la definición de interfaz validada, el Modelo de Ejecución define todas las características del producto software final dependientes del tipo de implementación de la Interfaz, así como el modo de comunicación entre interfaz y aplicación.

En este contexto, el Modelo Conceptual se apoya en el Diagrama de Acceso Navegacional (DAN). La información introducida en el DAN se almacena adecuadamente en un repositorio de información. A partir de él , y de acuerdo con el Modelo de Ejecución, un prototipo de Interfaz de Usuario funcionalmente equivalente puede ser construido de forma también

automática.

A continuación se presentan las características del Modelo Conceptual y se introduce el Modelo de Ejecución.

2.1 Modelo Conceptual

El modelado conceptual se realiza mediante el uso de uno o más Diagramas de Acceso Navegacional. Se necesita un mínimo de un DAN por cada tipo de agente al que se desee dar acceso al sistema. El punto de partida de los DAN es un diagrama de clases OO-Method simplificado, en el que se han seleccionado aquellas clases, atributos y métodos relevantes para ese tipo de agente. Este filtrado se basa tanto en las relaciones de Agente del Modelo de Objetos como en los Requerimientos de Navegación, que guían el diseño de las vistas de información necesarias para cada agente. Distintos DAN para un mismo usuario representan distintos prototipos de interfaz, y permiten su desarrollo incremental a medida que se modelan nuevas clases y métodos.

Vamos a introducir un ejemplo de sistema de información de una pequeña biblioteca que nos servirá para ir ilustrando los distintos conceptos de modelado del DAN del agente 'bibliotecario'. Este sistema está compuesto por socios y libros. Los libros se prestan a los socios como consecuencia de un préstamo. Un socio puede ser penalizado (considerado 'no fiable') si así lo decide el bibliotecario cuando la fecha del préstamo expira. A los socios no fiables no se les permite pedir prestados libros hasta que dejan de serlo.

Para diseñar el DAN se deben, en primer lugar, conocer los Requerimientos de Navegación (Casos de Uso de la interfaz) del agente involucrado en la vista del sistema que se pretende modelar. El DAN actuará como 'storyboard' abstracto de la vista del sistema que tiene el usuario considerado en cada caso.

En el ejemplo, se supondrá que los requisitos de navegación del sistema extraídos para un agente de tipo 'bibliotecario' son:

- Ver los préstamos actuales con dni del lector, nombre del libro prestado y fecha de préstamo.
- Ver los lectores (dni, nombre) que se han retrasado en la devolución del préstamo, y por cuántos días. Dar la posibilidad de penalizarlo.
- Ver los lectores penalizados y poder reconsiderar su posición de 'Lector poco fiable'.

Además el bibliotecario debe tener acceso a los servicios Añadir_Libro y Destruir_Libro de la clase Libro y Añadir_Lector y Borrar_Lector de la clase Lector.

Clases Navegacionales Las Clases Navegacionales (CN) son los componentes básicos del modelo, y se representan mediante un rectángulo con tres áreas (ver Figura 1):

- Cabecera: Nombre de la clase
- Atributos de la clase: parte estática donde se declaran los atributos relevantes para ese agente y esa vista.
- Servicios Navegacionales: parte dinámica donde se especifican los servicios activables por el agente vinculado a ese diagrama.

OO-$\mathcal{H}$Method define una Clase Navegacional como una Clase (en el sentido del modelado OO) enriquecida con características que la adaptan al contexto hipermedial, diferenciándola por tanto de la acepción que tiene en el resto de modelos.

Los atributos de la CN pueden ser de tres tipos, según la forma en la que son accedidos por los usuarios:

- Siempre visibles: su contenido se muestra siempre que el usuario visualice algún objeto de esa clase.
- Referenciados: su contenido se muestra referenciado mediante un enlace (o un mecanismo análogo) que indique su presencia, de manera que sólo si el usuario está interesado en ver su contenido activará dicho mecanismo.
- Ocultos: No se muestran en la página, y tampoco son referenciados. La única manera de visualizarlos es mediante una vista detallada (Zoom) del objeto. Es un modo de ocultar la información considerada poco útil (aunque no en todos los casos irrelevante) para ese usuario y el propósito general de esa aplicación.

El objetivo de esta clasificación es no desorientar al usuario con un exceso de contenido en las páginas finales.

Otro concepto importante aplicable a las CN es el de Perspectivas (P) [9,19]. La manera de capturar Perspectivas en el diagrama es definir atributos multivaluados. Un atributo multivaluado vendrá especificado por un nombre y un conjunto de perspectivas entre paréntesis. Las perspectivas se definen en el marco de una serie de ejes, cada uno representando una forma relevante de presentación de información. Por el momento se han incorporado al modelo tres de estos ejes:

- Lengua: inglés, francés, español, italiano.
- Nivel Discurso: simplificado, experto
- Medio: animación, imagen, texto.

De entre las perspectivas posibles se debe decidir cuál será la perspectiva por defecto, y señarlarla en el DAN mediante el signo +, del mismo modo que se hace en OOHDM [18]. A los distintos conjuntos de perspectivas se les puede dar un nombre y un ámbito, en cuyo caso se está definiendo una perspectiva-tipo. El ámbito de las perspectivas-tipo puede ser global al tipo de atributo o local al atributo considerado. Una perspectiva-tipo global a un tipo de atributo será heredada por todos los atributos de ese tipo. Como ejemplo imaginemos que el diseñador define el tipo 'descripción', y le asocia una perspectiva-tipo formada por dos estilos de presentación: (texto+, imagen), de ámbito global. De este modo se está indicando

que para todo atributo de cualquier clase cuyo tipo sea 'descripción' se definen de manera automática estas dos perspectivas.

Mientras que una perspectiva local añade formas de visualización a atributos locales, las perspectivas globales pueden ser entendidas como 'modos de visualización' [8] concurrentes en la aplicación, ya que proporcionan de forma general vistas distintas de la misma información.

Destinos Navegacionales Las CN se agrupan en Destinos Navegacionales (DN). Se define un DN como un conjunto de CN agrupadas para proporcionar una vista coherente del sistema. Se debe asociar un mínimo de un DN a cada Requerimiento Navegacional del usuario. Además, los DN tienen asociado un ámbito: local al tipo de usuario (y por tanto al DAN actual) o global al sistema.

La representación gráfica del DN (ver Figura 1) es un rectángulo que engloba a todas las clases navegacionales afectadas en esa vista. Un DN se identifica por un nombre, que se sitúa en una solapa situada en la esquina superior izquierda del rectángulo.

El concepto de DN que aparece en OO-$\mathcal{H}$Method ha sido capturado con ligeras variantes en otros modelos a través de la definición de nodos [19], entidades derivadas [9], hipervistas [7] o macroentidades [1], que engloban conjuntos de objetos con una presentación Web autónoma. La aportación de OO-$\mathcal{H}$Method en este ámbito consiste en basar la definición de los DN en el concepto de Requerimiento de Navegación, y no en la presentación física de la información.

Para OO-$\mathcal{H}$Method, el que la información se presente en una sola página web o por el contrario se divida en varias páginas es irrelevante en este estadio del diseño. De hecho un mismo DN puede tener varias materializaciones (páginas) distintas, de un modo similar a los 'targets' definidos en [4]. Así, la definición OO-$\mathcal{H}$Method de los DN captura implícitamente el patrón 'Contexto Navegacional' [17].

La selección de las distintas CN determina los objetos que se deben mostrar, y la definición de DN proporciona el contexto en que se presentan dichas clases. El siguiente paso es definir cómo navegar entre CN y/o DN. A la hora de definir la navegación hay que tener en cuenta toda una serie de aspectos ortogonales, como son la ordenación de objetos, filtrado, indexado o cardinalidad del acceso. Todo esto se contempla en los distintos elementos asociados a enlaces y colecciones, que se exponen a continuación.

Enlaces Navegacionales Un Enlace Navegacional (EN) se caracteriza conceptualmente por[1]:

- Nombre.
- Clase Navegacional origen.

[1] Patrones y Filtros serán presentados de forma exhaustiva más adelante en este mismo artículo

– Clase Navegacional destino.
– Patrón Navegacional.
– Filtros Navegacionales asociados.

En OO-$\mathcal{H}$Method se definen tres tipos de EN:

– Er: Enlace de requerimiento. Especifica el punto de entrada al destino navegacional. Todo destino tiene un punto de entrada, que se representa con un punto negro y una flecha apuntando a la clase navegacional raíz dentro de ese destino navegacional.
– Es: Enlaces de servicio. Enlace cuyo origen es una colección y cuyo destino es un servicio de una clase navegacional. Se representan mediante una flecha en forma de rayo. Cuando se utiliza un enlace de este tipo se especifica, además del acceso explícito del usuario al método, cómo dicho usuario debe introducir los parámetros del método y cómo va a visualizar los resultados.
– En: Enlaces de navegación. Dentro de ellos se distinguen dos tipos:
  • Ei: Enlaces internos: Son estructuras cuyo ámbito de actuación se encuentra dentro de un solo destino navegacional. Su característica básica es que su activación no cambia las características primordiales del contexto en el que se encuentra el usuario, por lo que no suelen introducir desorientación en el sistema.
  • Et: Enlaces de travesía. Se definen entre clases navegacionales pertenecientes a distintos destinos navegacionales. Supone habilitar caminos de navegación alternativos a los objetos de las clases destino involucradas.

Los EN son por definición enlaces estructurales[3], lo que posibilita su derivación automática [9] a partir del esquema conceptual. Además, son dirigidos. Esto implica que, si se desea navegar en dos sentidos, se deben especificar, explícita o implícitamente (a través de mecanismos como la derivación automática de enlaces), dos enlaces distintos. El título del enlace se utilizará como texto del mismo en la página Web generada.

Como se puede observar en la Figura 1, todos los tipos de EN tienen asociados 'Patrones Navegacionales' y 'Filtros Navegacionales'. A continuación se desarrolla cada uno de estos conceptos.

Patrones Navegacionales Un Patrón Navegacional (PN) se define como una solución recurrente a un problema de diseño de navegación. En algunos artículos se les denomina 'Patrones de Enlazado' [2].En OO-$\mathcal{H}$Method se definen cuatro Patrones Navegacionales asociados tanto a los Enlaces Navegacionales como a las Colecciones[2] Estos patrones han sido heredados de HDM-lite [7], y enriquecidos con una serie de atributos que completan el modo de navegación que representan:

– Índice: acceso a una lista de enlaces a los distintos objetos que conforman la población de la clase visualizada. En la vista de cada objeto existirá siempre un enlace al índice, además de los dependientes del patrón de navegación asociado a los distintos tipos de

[2] El concepto de Colección será presentado en el siguiente epígrafe.

relación que esa clase tenga con otras clases del sistema (agregación, herencia etc). Tiene asociado el atributo 'mostrar en origen/mostrar en destino', que determina si la lista de enlaces se incluirá en la página de la CN origen o si por el contrario se debe crear una nueva página, referenciada en la CN origen, que muestre los enlaces.
- Visita guiada: acceso secuencial a un conjunto de objetos de la población (cuáles sean esos objetos dependerá de los filtros navegacionales asociados al enlace). A cada vista de objeto se le añadirán las opciones siguiente, anterior, primero y último.
- Visita guiada indexada: combina el modo índice con el modo visita guiada. Tiene asociado el atributo 'mostrar en origen/mostrar en destino', que determina si el índice se debe incluir en la página de la CN origen o si por el contrario se debe crear una nueva página, referenciada en la CN origen, que muestre los enlaces.
- Showall: muestra todos los objetos de la población juntos en la misma página. Este modo lleva asociado el atributo 'cardinalidad', que permite paginar la respuesta (para limitar el tamaño de las páginas resultantes y por tanto el tiempo de espera del usuario). También tiene asociado el atributo 'mostrar en origen/mostrar en destino', que determina si la lista de enlaces se incluirá en la página de la CN origen o si por el contrario se debe crear una nueva página, referenciada en la CN origen, que muestre los enlaces.

La elección de patrón vendrá determinada por dos variables:

- Tipo de relación semántica que exprese el enlace navegacional
- Granularidad del enlace: cuántos objetos de la clase destino se prevé que estén relacionados con cada objeto de la clase origen.

Filtros Navegacionales Asociados a los EN aparecen los Filtros Navegacionales (FN). Un Filtro Navegacional captura una restricción sobre alguno de los atributos. Más concretamente, un FN restringe el orden, la cantidad o cualidad de los objetos de la CN destino. Por cantidad se entiende el número de objetos que conforman la población del destino, y por cualidad se hace referencia a los atributos navegacionales relevantes.

Se distinguen tres tipos de filtros:

- Filtros de atributo navegacional: especifican condiciones que deben cumplir los atributos de los objetos de la población destino.
- Filtros de condición: pueden representar parámetros de método (si se asocian a EN de servicio) o restricciones adicionales sobre la población destino. Un valor de filtro especificado con el signo dólar indica que dicho valor debe ser introducido por el usuario antes de poder atravesar el enlace al que está asociado. De este modo se pueden definir selectores de población arbitrarios.
- Filtros de orden: especifican el orden en que será accedida la población seleccionada.

A continuación se presenta una vista parcial del DAN del agente bibliotecario después de haber introducido los Enlaces de Requerimiento, de Navegación y de Servicio relevantes.

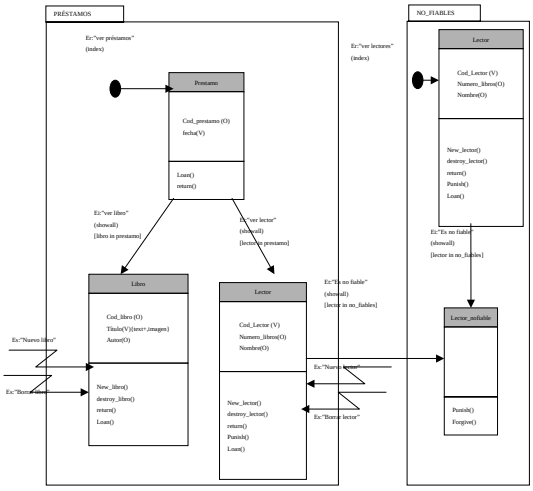

Figura1. *Especificación de los distintos tipos de enlace en el DAN del agente bibliotecario.*

Colecciones Otro aspecto importante de una Interfaz es la Navegación Externa, es decir, la forma de acceder a los distintos DN. Para ello se introduce un nuevo concepto: el concepto de colección. Una colección es una estructura, jerárquica o no, que abstrae determinados conceptos de la navegación externa. Las colecciones tienen un ámbito asociado (global, local al DN o local al CN). Son útiles para limitar las opciones de interacción entre usuario y aplicación, mejorando así la facilidad de uso de la misma. Un concepto parecido de colección aparece en diversas propuestas [6,4]. Una guía para el uso de colecciones es limitar en lo posible su profundidad para evitar causar desorientación al usuario.

OO-$\mathcal{H}$Method define 5 tipos básicos de colecciones:

- Clasificadores: definen estructuras de agrupación, jerárquicas o no, de información semánticamente relacionada. Son los más comunes.
- Punto Entrada. En la aplicación debe existir siempre una colección de este tipo. Su visibilidad es global y puede tener páginas estáticas asociadas.
- Punto de Salida: puntos por donde se puede salir de la aplicación y navegar por otros lugares Web. Al igual que el punto de entrada, pueden tener páginas estáticas asociadas.
- Menús: son agrupaciones, jerárquicas o no, de Enlaces de Servicio (si la colección agrupa Servicios Navegacionales) o de Requerimiento (si la colección agrupa distintos Destinos Navegacionales).
- Históricos: Existen diversas aproximaciones al concepto de 'histórico' [20], y OO-$\mathcal{H}$Method ha optado, en esta primera versión, por la más sencilla: un conjunto de los últimos x enlaces por los que se ha navegado, donde x es un parámetro de la colección, y donde la activación de uno de esos enlaces nos lleva al estado actual de la información requerida, no al estado que tenía cuando se visitó el enlace por primera vez.

Las colecciones (ver Figura 2), se representan mediante triángulos invertidos y tienen asociadas, al igual que los DN, reglas de visibilidad (globales, locales a un DN o locales a

una CN) y Patrones Navegacionales.

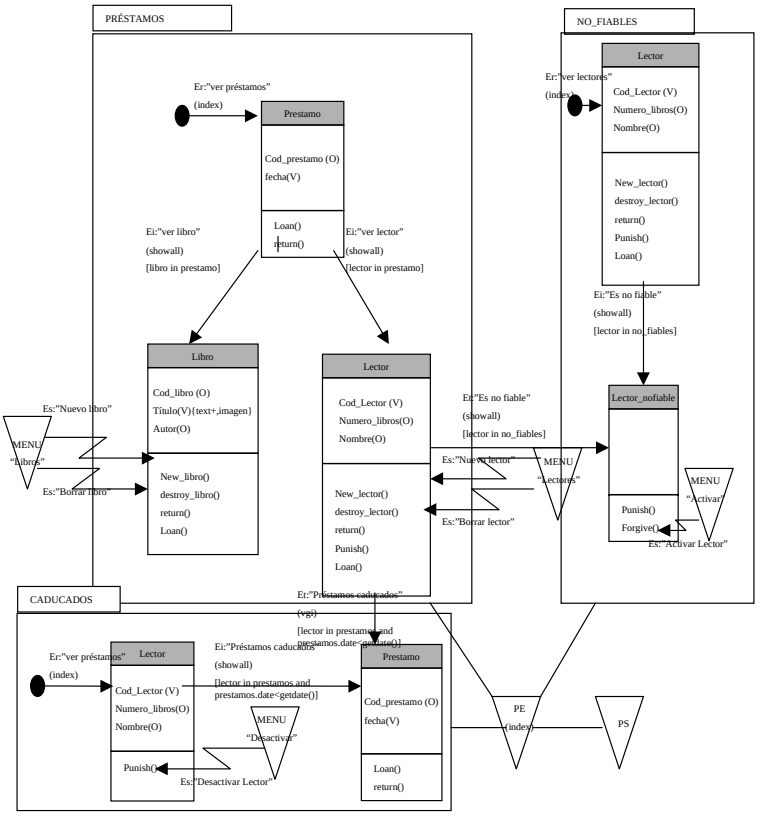

Figura2. *DAN simplificado del agente Bibliotecario.*

En la figura 2 se puede observar cómo quedaría el DAN del agente Bibliotecario para los requerimientos especificados. Se observa que el acceso a la función punish se realiza a partir del DN 'Caducados', correspondiente al segundo requerimiento. Esto, semánticamente, implica que antes de poder decidir penalizar a un lector el bibliotecario debe comprobar el estado de sus préstamos (debe estar en el contexto de Préstamos Caducados).

Plantillas El modo de determinar el aspecto que tendrá el sitio Web es mediante plantillas [6] que especificarán aspectos como situación física de los distintos ítems de información, tipografía [11], paleta de color, etc. de la población de cada destino navegacional y de cada colección. Existen numerosos argumentos en favor de la utilización de plantillas como complemento de metodologías de diseño[18].

Las plantillas cubren en OO-$\mathcal{H}$Method los aspectos relacionados con la capa de presentación de la Interfaz: describen tanto rasgos de visualización y contextualización (colores, elementos prediseñados que deben aparecer en la página, señalizadores que orientan al usua-

rio) como rasgos de estructuración (cómo repartir la información y los elementos visuales en la página).Además recogen, junto a los distintos Patrones de Interfaz, parámetros extraídos de experiencias anteriores de diseño. Mediante su uso se separa la capa lógica de navegación de la capa física de presentación de la Interfaz. Las plantillas además se agrupan formando una estructura arbórea. Así, ante la ausencia de definición de alguna característica básica en alguno de los niveles de presentación, se heredarán las características de la plantilla de nivel superior.

La construcción de la red de plantillas que conforma la interfaz Web en OO-$\mathcal{H}$Method se realiza en dos fases:

- Generación automática, a partir del DAN, de los esqueletos de plantilla asociados a cada entidad relevante (DN, CN, Colecciones) del modelo mediante una correspondencia simple, que se apoya en las implementaciones por defecto de los Patrones Navegacionales especificados en el DAN.
- Modificación, en caso de desearse un mayor nivel de sofisticación, de dichos esqueletos de plantilla para recoger particularidades de la vista diseñada.

2.2 Modelo de Ejecución

El Modelo de Ejecución proporciona los detalles de representación del Modelo Conceptual para un entorno de desarrollo concreto. Puesto que la estrategia de ejecución de la aplicación ya está definida en OO-Method, OO-$\mathcal{H}$Method se centra en determinar cómo implementar la información del Nivel de Interfaz asociada a ambientes Web. Esto implica el almacenamiento previo de las características de Interfaz reflejadas en el Modelo Conceptual en un repositorio (repositorio OO-$\mathcal{H}$Method) que proporcione la información necesaria para generar de manera automática la implementación en el lenguaje de programación deseado.

La definición de una 'Semántica de Navegación' por defecto (según las relaciones semánticas existentes en el esquema conceptual) y de una 'Apariencia de Interfaz' por defecto (plantillas que se elaborarán a partir de los distintos patrones de interfaz definidos) permiten:

- Sugerir de forma automática determinados enlaces
- Derivar de forma automática ciertos enlaces de aplicación
- Generar prototipos funcionales de la aplicación de forma automática.

3 Conclusiones

Las aplicaciones hipermediales modeladas con OO-Method y su extensión OO-$\mathcal{H}$Method son escalables, tienen un coste de mantenimiento reducido, su interfaz de usuario es dinámica y personalizable y sus enlaces son siempre consistentes.

OO-$\mathcal{H}$Method tiene un conjunto de características que lo diferencian de otros modelos, pero al mismo tiempo comparte muchos de los conceptos identificados como claves en el diseño de aplicaciones hipermediales, y por tanto se funda en la mayoría de los modelos

estudiados hasta el momento: HDM [9], HDM-lite [7], RMM [10], ADM [1] y, sobre todo OOHDM [19,18], con quien comparte su enfoque OO.

Desde el punto de vista de OO-$\mathcal{H}$Method una de las principales limitaciones de estos modelos es que se centran en sistemas hipermediales orientados a visualización y navegación a través de la información [3]. En este tipo de sistemas no existen (o al menos no se tienen en cuenta) mecanismos de interacción con el usuario aparte de sus decisiones de navegación. OO-$\mathcal{H}$Method, por el contrario, al extender las aplicaciones modeladas con OO-Method, proporciona mecanismos específicos para que el usuario interaccione con el sistema. La combinación de ambos modelos produce aplicaciones hipermediales que cubren tanto la estructura estática como el comportamiento.

Otra diferencia importante es que los modelos estudiados se centran desde el principio en los aspectos estructurales de la solución (datos y cómo van a ser presentados estos datos al usuario), en lugar de centrarse en los problemas estructurales del espacio del problema. Las aproximaciones en el espacio de la solución suponen períodos de desarrollo más largos y, en ocasiones, procesos de diseño más complejos. Nuestra aproximación parte de la premisa de que usuarios diferentes cambian en momentos diferentes la vista de un mismo esquema conceptual, por lo que la navegación a través de él debería ser independiente de dicho esquema. Esto es lo que hace OO-$\mathcal{H}$Method mediante la introducción del DAN, un diagrama que es exclusivamente navegacional, y en el que por tanto no se redefinen estructuras de información sino que el usuario se mueve de una vista del sistema a otra. La introducción de Patrones y Filtros Navegacionales ayudan a definir sin ambigüedad esta navegación.

4 Trabajos Futuros

En estos momentos OO-$\mathcal{H}$Method está en fase de definición y catalogación tanto de nuevos Patrones de Navegación como de Patrones de Interfaz suficientemente generales como para garantizar la reutilización de código. Los Patrones de Interfaz se integrarán en la estructura de las plantillas y proporcionarán una manera homogénea de acceder a objetos con determinadas características. Un ejemplo de Patrón de Interfaz es el patrón de Introducción de Tipos, que determina las validaciones que hay que realizar sobre cada campo y la forma en que se debe presentar dentro de un formulario de introducción al usuario.

Otra línea de trabajo importante es la generación de un nuevo Diagrama de Presentación que capture tanto las plantillas como los Patrones de Interfaz de manera gráfica, y por tanto mucho más intuitiva de cara al diseñador.

En todos los casos el estudio de la facilidad de uso de los patrones y técnicas elegidas está siendo un factor crítico para su incorporación en el modelo OO-$\mathcal{H}$Method.

Referencias

1. P. Atzeni, G. Mecca, and P. Merialdo. Design and Maintenance of Data-Intensive Web Sites. In *Advances in Database Technology - EDTB'98*, pages 436–449, 03 1998.
2. M. Bernstein. Patterns of Hypertext. In *HYPERTEXT '98. Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space. Structure in hypermedia systems*, pages 21–29, 1998.
3. M. Bieber and C. Kacmar. Designing Hypertext Support for Computational Applications. *CACM: Communications of the ACM*, 38(8):99 – 107, 1998.
4. S. Ceri, P. Fraternali, and S. Paraboschi. Design Principles for Data-Intensive Web Sites. *SIGMOD Record*, 28:84–89, 03 1999.
5. E. M. Fayad and D. C. Schmidt. Object-oriented application frameworks. *CACM: Communications of the ACM.*, 40(10):32–38, 10 1997.
6. F. M. Fernández, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of ACM SIGMOD International conference on Management of data*, pages 414–425, 10 1998.
7. P. Fraternali and P. Paolini. A Conceptual Model and a Tool Environment for Developing more Scalable, Dynamic, and Customizable Web Applications. In *Advances in Database Technology - EDBT'98*, pages 421–435, 1998.
8. F. Garzotto, L. Mainetti, and P. Paolini. Designing Modal Hypermedia Applications. In *Proceedings of the eight ACM conference on HYPERTEXT '97*, 1997.
9. F. Garzotto and P. Paolini. HDM A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems (TOIS)*, 11(1):1–26, 01 1993.
10. T. Isakowitz, E. A. Stohr, and V. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. *CACM: Communications of the ACM.*, pages 34–44, 08 1995.
11. P. Kahn and L. Krzysztof. Principles of Typography for User Interface Design. *Interactions of the ACM*, pages 15–29, 11 1998.
12. D. B. Lowe, J. A. Bucknell, and G. R. Webby. Improving Hypermedia Development: A Reference Model-Based Process Assessment Method. *Proceedings of the 10th ACM Conference on Hypertext and hypermedia: returning to our diverse roots.*, pages 139–146, 1999.
13. G. Mecca, P. Merialdo, P. Atzeni, and V. Crescenzi. The ARANEUS Guide to Web-Site Development. Technical report, Università di Roma, 03 1999.
14. M.Ñanard, J.Ñanard, and P. Kahn. Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates. In *HYPERTEXT '98. Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems*, pages 11–20, 1998.
15. O. Pastor, E. Insfrán, V. Pelechano, J. Romero, and J. Merseguer. OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods. In *CAiSE '97. International Conference on Advanced Information Systems*, pages 145–158, 1997.
16. O. Pastor, V. Pelechano, E. Insfrán, and J. Gómez. From Object Oriented Conceptual Modeling to Automated Programming in Java. In *ER '98. International Conference on the Entity Relationship Approach*, pages 183–196, 1998.
17. G. Rossi, D. Schwabe, and A. Garrido. Design Reuse in Hypermedia Applications Development. In *Proceedings of the eight ACM conference on HYPERTEXT '97*, pages 57–66, 1997.
18. D. Schwabe and R. Almeida Pontes. A Method-based Web Application Development Environment. In *Position Paper, Web Engineering Workshop, WWW8*, 1999.
19. D. Schwabe, G. Rossi, and D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proceedings of the the seventh ACM conference on HYPERTEXT '96*, page 166, 1996.
20. L. Tauscher and S. Greenberg. Revisitation patterns in World Wide Web navigation. In *CHI '97. Proceeding of the CHI 97 conference on Human factors in computing systems*, pages 399–406, 1997.