Departamento de Lenguajes y Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla

Avda Reina Mercedes, s/n. 41012 SEVILLA
Fax : 95 455 71 39. **Tlf**: 95 455 71 39. **E-mail**: lsi@lsi.us.es

Metodologías para el desarrollo de sistemas de información global: análisis comparativo y propuesta

Doctorando: María José Escalona Cuaresma
Directores: Manuel Mejías Risoto
Jesús Torres Valderrama

Sevilla, Octubre de 2001

HOJA DE CONTROL DE MODIFICACIONES

Número	Fecha	Descripción	Autores
0	10-2001	Desarrollo Inicial	M.J.Escalona

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

Hoy en día las aplicaciones que se desarrollan son muy diferentes a las que se desarrollaban hace unos años. El desarrollo de las comunicaciones y la gran difusión de Internet han creado nuevas necesidades dentro de los sistemas software.

En este sentido, muchas definiciones de sistemas se están dando dentro del mundo de la ingeniería del software: sistemas multimedia, sistemas hipermedia, aplicaciones web, sistemas de información global, etc.

Este trabajo es el resultado de la investigación que se ha llevado a cabo para estudiar una posible metodología para el desarrollo de sistemas de información global. En él se hace un recorrido de todas las opciones que se están aplicando en este ámbito y se presenta los primeros inicios de una nueva propuesta que actualmente se está elaborando.

Sin embargo, antes de comenzar a plantear el problema que nos va a ocupar el contenido de este trabajo, es necesario enmarcar el entorno de trabajo en el que nos vamos a mover. Para ello, es necesario hacer una serie de definiciones previas.

Desde que el desarrollo de aplicaciones informáticas se empezó a considerar un proceso de ingeniería, muchas metodologías de desarrollo han ido naciendo con el fin de dar soporte al ciclo de desarrollo del proyecto. Entre estas, podemos destacar algunas como MÉTRICA, MERISE, SSADM y ya más recientes como OMT o el actual UML. Todos estas metodologías estaban orientadas desde sus comienzos al desarrollo sistemas para gestionar una información que se encuentra almacenada en una o varias bases de datos, distribuidas o no. Entre los aspectos que se tratan como más críticos en estas metodologías, los más importante son el almacenamiento y la recuperación adecuada de la información, así como que las posibilidades funcionales que ofrezcan sean las necesarias.

Otro término que comienza a nacer es el de las *aplicaciones multimedia*. Mientras que las metodologías anteriores tratan con especial interés los aspectos de almacenamiento y funcionalidad, en las aplicaciones multimedia el objetivo esencial va a ser difundir información almacenada en diferentes medios (imágenes, vídeos, música, etc.) de manera que llegue al público no experto en informática de una forma sencilla, fácil e intuitiva. Son aplicaciones que nos preocupan tanto por las necesidades de almacenamiento, puesto que en muchos casos no tienen ni bases de datos asociadas, y la funcionalidad, como por la apariencia y el interfaz del sistema. El desarrollo de estos sistemas comienzan a realizarse sin que haya ninguna norma que se pueda tomar como marco de referencia para su desarrollo. Sin embargo, a principios de los 90, se comienza a estudiar la necesidad de una metodología que guíe a los desarrolladores y que asegure la calidad de los productos multimedia generados. Por esta razón, desde el año 93 comienzan a publicarse propuestas metodológicas y nuevos modelos para representar la problemática de estas aplicaciones: HDM [Garzoto 1993], RMM [Isakowitz 1995], EORM [Lange 1995], OOHDM [Rossi 1996], etc.

Junto con todo esto, Internet va tomando cada vez más popularidad. Hoy en día Internet supone un excelente medio para obtener información de los más variados temas sin necesidad de movernos de casa. Internet nos permite, a través de interfaces sencillas, intuitivas y amigables, hacer desde una consulta a la cartelera de cine hasta comprar acciones en nuestra entidad bancaria. Sin embargo, esto no siempre ha sido así. En sus comienzos la Web nació como una red que conectaba a varias redes privadas, públicas y

de centros de investigación, con el objetivo de compartir información entre grupos de investigación localizados en diversos lugares.

Cuando la red de redes comenzó a tomar popularidad, las páginas que se mostraban en los servidores de las organizaciones eran páginas estáticas que podríamos describir como documentos mostrados a través de Internet.

Sin embargo, poco a poco y, a medida que va aumentando la popularidad de la red, las empresas y organizaciones comienzan a requerir que sus páginas tengan un cierto dinamismo que las haga más atractivas para el usuario. Este dinamismo, se introduce haciendo que las páginas sean cada vez más interactivas, permitiendo que el usuario no solo sea un sujeto pasivo en la página, sino que, comienza a interactuar mandando datos al servidor remoto y recibiendo respuestas de este.

Poco a poco Internet adquiere popularidad y las organizaciones ven en ella una posibilidad de publicidad efectiva y de facilitar la comunicación con sus clientes. Comienza así la necesidad de incluir bases de datos y grandes volúmenes de información en la red.

A finales de los 90, las páginas Web han dejado de ser páginas documentales puramente estáticas para dejar paso a páginas interactivas, en las que se incluye información en múltiples medios y con las que el usuario puede interactuar. En estas páginas se han unido los conceptos de hipertexto con los múltiples medios, dando origen a lo que se conoce como *hipermedia*. Además, la nueva generación de páginas Web está dotada de una gran funcionalidad. Ya no tenemos páginas que muestran y recogen datos, lo que actualmente se tiende a mostrar por Internet son complejas aplicaciones con grandes requisitos de almacenamiento y funcionalidad, acompañado de un complejo y llamativo interfaz, que engloba información en múltiple medios y enlaces no solo entre datos textuales, sino que, permite también enlaces entre datos multimedia. Son los llamados *sistemas de información web*, que están dando origen a complejos portales que se están ganando la atención del público actual.

Debido a este gran avance y promovidos por las mismas inquietudes de otros entornos, desde el año 98 están empezando a nacer metodologías que van orientadas a la web y al desarrollo de estos sistemas de información. Las metodologías para la web nacen en la mayoría de los casos de otra metodología anterior, bien de una metodología clásica, como el caso de Conallen [Conallen 1999], basada en el Proceso Unificado y UML; o bien de una metodología para aplicaciones multimedia, como el caso de los nuevos desarrollos de OOHDM [Schwabe 2001].

Otro término importante que aparece también hoy en día es el concepto de *biblioteca digital*. Una biblioteca digital (BiDi) es una biblioteca que ha sido extendida y mejorada mediante la aplicación de la tecnología digital. Es la unión de ordenadores, sistemas de almacenamiento y redes de comunicaciones con el contenido y el software necesario para reproducir, emular y extender los servicios proporcionados por las bibliotecas convencionales. Una biblioteca digital debe cumplir todas las tareas de una biblioteca convencional y explotar las ventajas de la tecnología digital en el almacenamiento, la búsqueda y las comunicaciones, además de la integración de nuevos tipos de medios (textos, imágenes, sonidos, vídeos, animaciones, etc). La biblioteca digital proporciona a una comunidad de usuarios un acceso coherente a repositorios de información grandes y organizados. Las bibliotecas digitales son construidas (recogiendo y organizando la información) por una comunidad de usuarios y sus funcionalidades son acordes a las necesidades de información de dicha comunidad. Las posibilidades de los usuarios para acceder, reorganizar y utilizar este repositorio están

enriquecidas con las capacidades de la tecnología digital. También han aparecido propuestas para el desarrollo de aplicaciones basadas en BiDi [Escalona 2000a].

Como resumen a esto, podríamos decir que durante los últimos años han venido naciendo diferentes metodologías que se mueven en entornos concretos dando más importancia a los aspectos más importantes de los sistemas que tratan. En la multimedia, el tratamiento de los múltiples medios y el entorno visual es esencial, de ahí que las metodologías de la multimedia se centren en estos aspectos. Las metodologías de la web, van a estar más orientadas a aspectos de navegación y arquitecturas. Mientras que el resto de las metodologías más generales, el Proceso Unificado, OMT, etc. van a centrarse más en aspectos como el almacenamiento de la información y la funcionalidad.

En la actualidad, las aplicaciones se desarrollan normalmente en entornos distribuidos, es muy común el que se distribuyan por internet y normalmente tienen asociados elementos multimedia e hypermedia en grandes bases de datos. Se caracterizan por tener grandes requisitos funcionales y de seguridad, múltiples usuarios y en muchos casos indefinidos y con diferentes grados de conocimiento. Estas aplicaciones se conocen como *sistemas de información global*, y son, por decirlo de alguna manera, un concepto mucho más genérico que engloba a las aplicaciones que se encuentran en los otros grupos.

El sistema de información global puede verse como si fuera una aplicación multimedia, puesto que normalmente maneja información almacenado en múltiples medios. Pero cuando se distribuye a través de internet, se podría ver como un sistema de información web. Sin embargo, ninguna de las metodologías de estos ámbitos sería adecuada, puesto que no tratan los aspectos de almacenamiento y funcionalidad de manera adecuada, que en los sistemas de información web suelen ser bastante críticos.

Los sistemas de información global almacenan grandes cantidades de información y requieren sistemas de seguridad muy potentes, así como una funcionalidad muy elaborada que asegure que los usuarios van a poder trabajar con esta información de manera adecuada. Para tratar estos aspectos, metodologías como el Proceso Unificado podrían ser un buen marco de referencia para su desarrollo. Por otro lado, y cuando su medio de transmisión es la web, cosa que suele ser muy común, el sistema de información global adquiere todas las características de un sistema de información en la web. Normalmente, estos sistemas deben tener una interfaz intuitiva y amigable que hacen uso de la multimedia para llegar más fácilmente al usuario. Así un sistema de información global podría verse como una aplicación multimedia. Pero además si analizamos la definición de biblioteca digital, ¿no podríamos decir que un sistema basado en bibliotecas digitales es un sistema de información global?.

Por ser el sistema de información global el más genérico, este trabajo se mueve en este entorno. Aquí vamos a estudiar las propuestas metodológicas definidas para sistemas multimedia, sistemas de información web, etc. Con el fin de encontrar modelos e ideas consensuadas que nos permitan desarrollar una metodología para el desarrollo de los sistemas de información global.

2. Planteamiento del problema

Ya hemos analizado en el ámbito que nos vamos a mover: los sistemas de información global. El problema que se plantea resolver es el de elaborar un marco metodológico para el desarrollo de este tipo de sistemas.

Para ello, antes es necesario estudiar qué debe ofrecer una metodología para el desarrollo de sistemas de información global.

Un sistema de información global posee, como ya se ha comentado, grandes bases de información, importantes y complejos requisitos funcionales y deben ser seguras y eficaces. Debe permitir el trabajar con la información multimedia y ofrecer una interfaz amigable y adaptada al usuario, por la que sea fácil navegar[1], como los sistemas multimedia. Pero, además, un sistema de información puede desarrollarse en muchos entornos arquitectónicos diferentes: la web, bases de datos federadas, etc.

Una metodología para el desarrollo de sistemas de información global, debe ofrecer las herramientas y técnicas suficientes como para cubrir todos esos aspectos que se pueden encontrar en un sistema de este tipo. Así, en la figura 1 encontramos un esquema con las características que debe contemplar y estudiar el proceso metodológico de la propuesta.

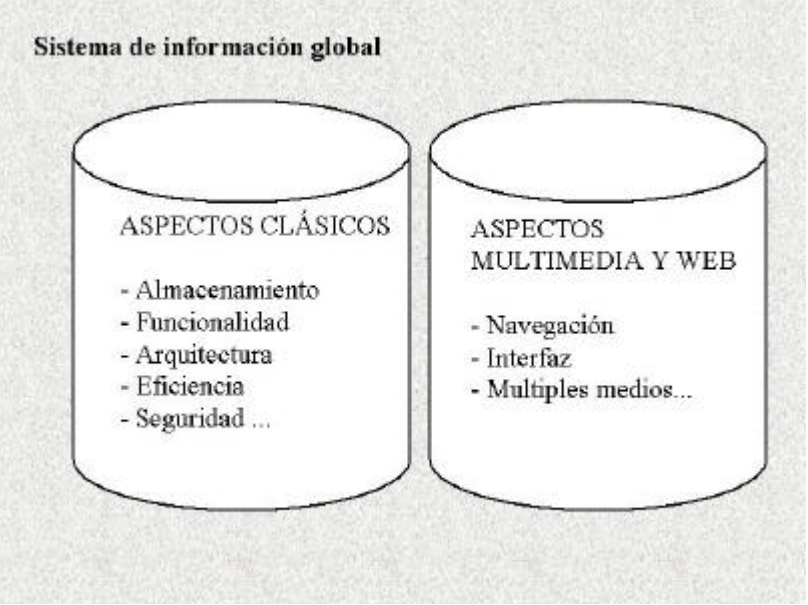

FIGURA 1: ASPECTOS DE LOS SISTEMAS DE INFORMACIÓN GLOBAL

Pero, ¿qué debe ofrecer la metodología?. En el apartado 5 de este trabajo, se analizarán diferentes propuestas metodológicas para diferentes entornos. Ninguna de ellas va a adecuarse al cien por cien a los sistemas de información global, principalmente porque están orientados a un tipo de sistema concreto. Pero otro problema que vamos a observar muy a menudo es que estas metodologías están agrupadas en tres grandes grupos:

- Las orientadas al proceso, indican qué secuencia de pasos hay que seguir para el desarrollo del sistema.

[1] El concepto de navegación es más genérico que el propio de internet. Navegación en el entorno de los sistemas de información global se refiere a la posibilidad de manejar el cambio de pantallas del sistema.

- Las orientadas al producto, que se preocupan esencialmente por lo que hay que entregar al realizar las fases.
- Las orientadas a las técnicas, son propuestas que se preocupan por exponer nuevas técnicas, o ampliar las ya existentes, para adecuarlas a las necesidades que existan.

Una propuesta metodológica para el desarrollo de sistemas de información global debe abordar las tres orientaciones puesto que debe indicar al desarrollador qué hacer, cómo hacerlo y qué debe conseguir.

3. Relevancia del problema

Desde hace ya tiempo, el desarrollo de los sistemas software se ve como un proceso de ingeniería que dio origen a la ingeniería del software. El desarrollo de un sistema de información global es una tarea complicada. Normalmente requiere tener un amplio conocimiento informático, puesto que sus necesidades de información, funcionalidad y difusión suelen ser muy complejas. Pero además es necesario que en el equipo de desarrollo haya personas que sean expertas en la información y el entorno de trabajo en el que se mueve la aplicación. Por ello, se debe ofrecer un marco de desarrollo, que sea lo suficientemente completo como para dar soporte a los desarrolladores informáticos y a la vez suficientemente intuitivo como para que, al menos en las primeras fases del desarrollo, sea fácil de entender por los expertos en la materia.

Los sistemas de información global están adquiriendo una gran importancia y a la hora de su desarrollo el equipo de trabajo se encuentra con el problema de qué metodología usar. Las metodologías actuales, tanto el marco de los sistemas multimedia como el marco de la web, no contemplan todas las necesidades de estos sistemas con la profundidad adecuada. Es necesario trabajar los aspectos de navegación e interfaz, la multiplicidad de los medios, las necesidades de almacenamiento y las otras características que vimos en la figura 1 desde las primeras fases del ciclo de vida. Además es necesario ofrecer un marco de desarrollo que indique qué hacer, cómo hacerlo y qué presentar en cada momento para asegurar un producto que se adecue a las necesidades y que sea de fácil mantenimiento, en definitiva que sea de calidad.

El objetivo principal pues de este trabajo, es sentar las bases para desarrollar un sistema de información global de calidad.

4. Aspectos resueltos y por resolver

Este trabajo es una introducción a un proyecto cuya idea es la de plantear una metodología de desarrollo para los sistemas de información global, tal y como se ha venido diciendo en los apartados anteriores.

El proceso de trabajo que se está siguiendo en dicho proyecto es el que se muestra en la figura 2.

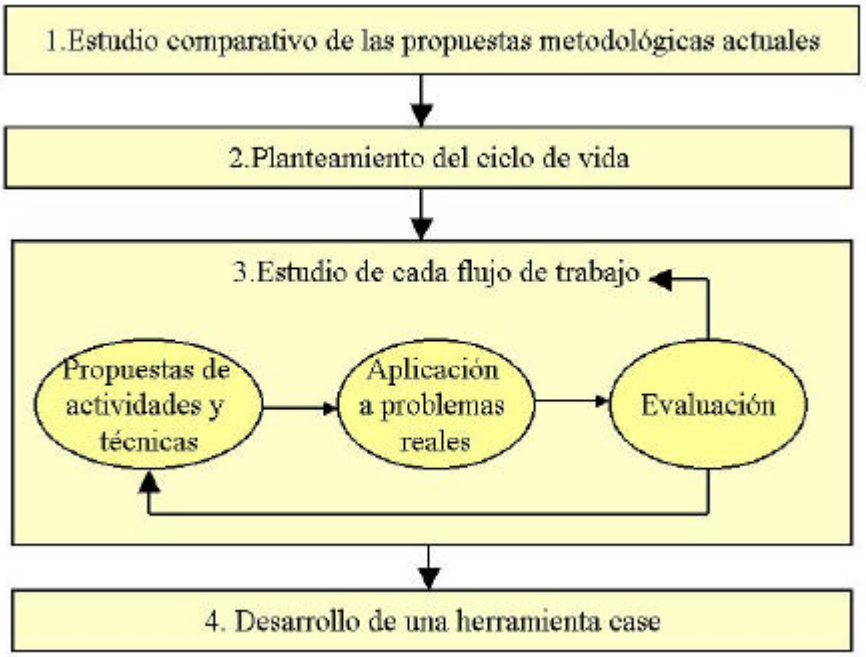

FIGURA 2: PROCESO DE TRABAJO

La primera tarea a realizar fue estudiar qué propuestas metodológicas se podrían usar en el desarrollo de un sistema software. Tras este estudio, cuyo resumen se presenta en el apartado 5 de este trabajo, se concluyo qué:

1- Las metodologías actuales no cubrían todos los aspectos que requieren estos sistemas.

2- Estas propuestas no ofrecían un marco que indicara el proceso a seguir, las técnicas a aplicar y los productos a obtener.

3- La mayoría de ellas coinciden en muchos aspectos que son adecuados y que hay que tener en cuenta para la propuesta de nuestra metodología y que estudiaremos al final del apartado 5.

Tras destacar las necesidades no cubiertas y encontrar aquellos puntos válidos que parecían buen soporte para nuestra propuesta. La segunda etapa fue el proponer un ciclo de vida que fuese adecuado para nuestra propuesta, éste se presenta en el apartado 6 de este trabajo.

Este ciclo de vida, como se verá se divide flujos de trabajo. Por ello, tras este planteamiento se comenzó a analizar cada uno de ellos. La idea para estudiar cada uno de ellos es

1- Realizar un planteamiento de las actividades que habría que seguir en el mismo y de las técnicas que los desarrolladores deben aplicar para conseguir los productos.

2- Una vez planteado esto, se realizan pruebas en sistemas reales para ver si la aplicación de la propuesta es adecuada.

3- Por último se realiza una evaluación. Si es adecuada se pasa al siguiente flujo y si no se revisa el paso 1 comenzando de nuevo el ciclo.

Actualmente se ha definido el primer flujo de trabajo y el segundo flujo se encuentra en el paso número 1. En total son 5 flujos, con lo que la metodología aún está en sus comienzos.

En el apartado 6 se presentan estos flujos, el primero y el segundo con mucho más detalle. La propuesta que se está elaborando se define por ser muy metódica y estructurada por lo que ofrece la posibilidad de generar una herramienta case que facilite el proceso de desarrollo. Esta es la última fase que se pretende abordar.

5. Comparativa de propuestas

5.1.Introducción

Dentro de la variabilidad que ofrecen los sistemas de información global, existen muchas tendencias de metodologías que ofrecen diferentes marcos que los desarrolladores pueden asumir a la hora de realizar su trabajo.

Estos trabajos están orientados a diferentes entornos de trabajo: aplicaciones multimedia, aplicaciones en la web, bibliotecas digitales, etc.

Un sistema de información global, como se ha destacado en los apartados anteriores, es un concepto mucho más general que engloba a todos estos sistemas. Si se desea proponer una metodología de desarrollo lo suficientemente genérica y a la vez precisa como para que permita modelar de forma adecuada todos los aspectos propios de cada sistema, es necesario hacer un estudio de las tendencias que actualmente se encuentran en vigor en cada uno de estos ámbitos. Por ello, a continuación se hará un análisis de las principales propuestas y se introducirán aquellos puntos más relevantes de cada una de ellas.

Para facilitar la compresión de mucho de los modelos que se explicarán, en el punto 5.2 se detalla un ejemplo que será utilizado en aquellas ocasiones que se estimen oportunas.

Antes de comenzar con dicho ejemplo y con el estudio comparativo, es necesario introducir el ámbito en el que nos movemos. Se van a presentar un total de quince propuestas. Alguna de ellas se mueven en el entorno de la multimedia, otras en el entorno de la web y otras son mucho más genéricas.

En la tabla 1 se muestra un esquema en el que se recoge el nombre de la propuesta, la referencia al trabajo en el que se presenta, así como el año en el que se desarrolló. En la tercera columna se recoge el ámbito de trabajo en el que se enmarca. El orden que se ha seguido para la presentación de estas propuestas ha sido el cronológico. Se ha elegido este orden de presentación porque muchas propuestas se basan en propuestas anteriores y así el lector conocerá los conceptos que se referencien.

Por último, indicar antes de empezar que a la hora de realizar este trabajo se ha presupuesto un cierto conocimiento en algunas propuestas estándar. Muchas de las metodologías que se van a estudiar hacen uso de modelos como los ERDs, diagramas de clases, diagramas de casos de uso, etc. y de lenguajes de modelado como UML o las propuestas de OMT, que no se explican en este documento.

Nombre de la propuesta	Referencia	Ámbito
HDM	[Garzoto 1993]	Multimedia
RMM	[Isakowitz 1995]	Multimedia
EORM	[Lange 1995]	Multimedia
The MacWeb Hypermedia Design Method	[Nanard 1995]	Web, aunque muy orientado a la multimedia
OOHDM	[Rossi 1996]	Multimedia, aunque sus últimas versiones están orientadas a la web
WSDM	[De Troyer 1997]	Web
OO-Method; OO-HMethod	[Pastor 1997]	Trabaja con aplicaciones de gestión en general, aunque con el nacimiento de OO-HMethod se está orientando a la web
SOHDM	[Lee 1998]	Multimedia
RNA	[Bieber 1998]	Web
HFPM	[Olsina 1998]	Web y Multimedia
OO/Pattern Approach	[Thomson 1998]	Web y Multimedia
Proceso Unificado de UML	[Jacobson 1999]	Trabaja con sistemas de gestión en general
Building Web Applications with UML	[Conallen 1999]	Web
Specification and modeling of multimedia and hypermedia systems	[Mandel 2000]	Web y Multimedia
A UML-Based methodology for Hypermedia Design	[Hennicker 2001]	Web y Multimedia

TABLA 1: RESUMEN DE LAS PROPUESTAS

5.2.Ejemplo

Como ya se ha enunciado, para aplicar algunos modelos y técnicas, se va a plantear un ejemplo real. Se pretende desarrollar un sistema para la gestión y difusión de los datos del patrimonio cultural de Andalucía. En concreto del patrimonio mueble.

Este problema lo estamos tomando como caso de aplicación en nuestro grupo de investigación para poner en práctica nuestro trabajo, que se detallará en el apartado 6 de este documento que ha servido como base en varias publicaciones.

Como la dimensión de este sistema es bastante amplia, vamos a abordar solamente una parte del mismo. Se pretende, pues, realizar un sistema que permita gestionar y catalogar el patrimonio mueble de Andalucía, teniendo en cuenta que para un bien guardamos: su código, que será un número propio de cada bien; su denominación; su acceso, o lo que es lo mismo si es público o privado; la cronología o fecha en la que se data; el grado de certeza de esta cronología y una serie de observaciones. Además se almacenarán una serie de imágenes del bien; los períodos históricos en los que se ubica y los estilos en los que se engloba; sus autores y las tipologías de piezas en las que se puede ubicar (cuadro, escultura, etc.). Por último se deberá conocer el inmueble o edificio en el que se ubica habitualmente y el que se ubica de forma temporal (por restauraciones, exposiciones, etc.). Por cada inmueble almacenaremos su dirección y su nombre.

A la hora de presentar la información, se presentará en dos grupos diferentes. Un primer grupo en el que tendremos los datos de identificación, en el que se engloba el código, la denominación, la ubicación habitual y la temporal, el acceso y las observaciones. Y otro en el que se mostrarán los datos de descripción y que recogerá las

imágenes, las tipologías, los estilos, los autores, los períodos históricos, la cronología y la certeza.

A este sistema pueden acceder diferentes actores para trabajar con él. Estos actores los podemos clasificar usando dos criterios:

?? Su perfil investigador, en cuyo caso pueden ser arqueólogos, artistas, etnólogos o varios de estos a la vez. Dependiendo del tipo de investigador se le mostrarán unos datos u otros. Así la cronología y la certeza, y los autores, no tienen sentido para los arqueólogos.

?? Su perfil de usuario, en cuyo caso pueden ser:

- Un usuario del centro, que es el que tiene más permisos. Pueden dar de alta, modificar o consultar cualquier información.
- Un catalogador, que podrá dar de alta, modificar y consultar sólo los bienes de la provincia o municipio en el que sea catalogador.
- Un usuario externo, que puede consultar cualquier información pero nunca dar de alta o modificarla.

Como puede verse, no se ha descrito si este sistema será distribuido por la web o no o si es un sistema clásico o multimedia. Esto facilita el hecho de que podamos usarlo dentro de todas las propuestas que vamos a utilizar.

5.3.Estudio de las diferentes propuestas

Pasemos ahora a analizar cada una de las propuestas que se han resumido en el punto 5.1 de forma más detallada. Para cada una de ellas se hará una descripción breve, se detallarán sus fases, cuando las haya, se aplicará el ejemplo a la parte que se estime oportuno y se introducirán una serie de conclusiones sobre las mismas.

5.3.1. HDM- A Model-Based Approach to Hypertext Aplication Design

HDM (Hypermedia Design Model) [Garzotto 1993] es uno de los primeros métodos desarrollado para definir la estructura y la navegación propia de las aplicaciones multimedia. HDM se basa en el modelo Entidad-Relación, aunque amplía el concepto de entidad e introduce nuevos elementos, como las unidades o los enlaces.

En HDM se pretende especificar la aplicación mediante un modelo Entidad-Relación extendido. Este modelo va a representar la estructura global de la aplicación sin entrar en detalles de desarrollo de los elementos unitarios (nodos de la aplicación).

A pesar de que en la actualidad HDM no se usa, ha servido como base a otras importantes metodologías como son RMM [Isakowitz 1995] y OOHDM [Schwabe 1995], las cuales veremos más adelante.

1.Conceptos básicos de HDM

HDM propone un conjunto de elementos que permiten al diseñador especificar una aplicación. Estos elementos son las *entidades*, los *componentes*, las *perspectivas*, las *unidades* y los *enlaces*. Todos estos elementos pueden incorporarse en la semántica del clásico modelo Entidad-Relación. Sin embargo, y a pesar de que términos como las entidades hayan sido heredados de los ERD veremos a continuación que han sido extendidos para poder representar una estructura compleja que contenga enlaces y una semántica de navegación interna.

En definitiva una aplicación especificada mediante un modelo HDM consiste en una estructura general compuesta por unas unidades básicas denominadas entidades. Una entidad denota un objeto físico o conceptual del universo de discurso de la aplicación. En HDM las entidades son agrupadas en *tipos de entidad*. Los tipos de entidad se caracterizan por un *nombre*, por un conjunto de *perspectivas* bajo las que se pueden presentar su contenido y un conjunto de *enlaces de aplicación* por los que se puede navegar (estos conceptos se verán más adelante). Una entidad es la unidad mínima autónoma de cualquier modelo HDM, pero existen otros conceptos añadidos que veremos a continuación.

Cada entidad está compuesta por una jerarquía de *componentes* que heredan las propiedades de dicha entidad. Los componentes no tienen razón de ser sin que exista la entidad de la que dependen. Los componentes son, por su parte, abstracciones para diseñar un conjunto de *unidades* o *nodos* que representan un mismo conjunto de información de la entidad. Una unidad, es pues un depósito de la información contenida en una aplicación. Una unidad representa un fragmento del contenido de una entidad presentada bajo una *perspectiva* particular. De esta forma, la perspectiva permite representar la multiplicidad de presentaciones de un mismo contenido de información (por ejemplo, la presentación de un documento en múltiples lenguas). Para entender mejor la relación entre estos elementos, usaremos la potencia representativa de UML [Jacobson 1999]. Vemos así en la figura número 3 un modelo de clases que representa las relaciones entre los elementos de HDM.

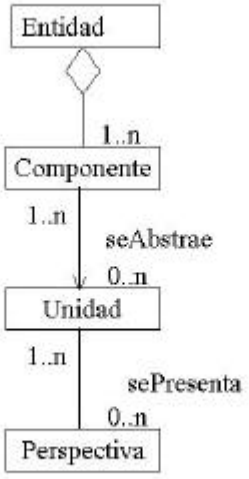

FIGURA 3: RELACIONES DE LOS ELEMENTOS DE HDM

En un modelo HDM las estructuras de información pueden ser conectadas mediante *enlaces.* Un enlace entre dos elementos indica que en la aplicación hipermedia resultante existe la posibilidad de *navegar* entre esos dos elementos. HDM distingue tres tipos de enlaces:

- Enlaces estructurales, conectan componentes de la misma entidad.
- Enlaces de perspectiva, conectan perspectivas que corresponden a una misma unidad.

✍✍Enlaces de aplicación, sirven para conectar componentes y unidades. Son los más complejos de los tres pues pueden conectar elementos de unidades diferentes. Los enlaces de aplicación se organizan en *tipos de enlace de aplicación* o simplemente *tipos de enlace*. Los tipos de enlaces se caracterizan por un *nombre*, un conjunto de *fuentes*, que indica de dónde puede partirse en la navegación, y un *destino*, que indica hacia dónde va el enlace. Por último tiene un atributo especial que puede tomar los valores *simétrico* o *asimétrico*, para indicar si el enlace es en un único sentido o no.

Como cualquier otro modelo de diseño, HDM distingue entre el concepto de *esquema* y de *instancia del esquema*. El esquema define la estructura general de la aplicación y la instancia son las unidades, perspectivas y enlaces concretos que cumplan con los requisitos establecidos en el modelo.

En una aplicación multimedia es necesario plantearse, además, lo que HDM define como *semántica de navegación*. La semántica de navegación representa cómo se va a mostrar la información al usuario. Para ello, HDM define una semántica de navegación por defecto. En ésta se asume que al usuario se le muestra la información mediante nodos o unidades de forma que sólo un nodo está activo en cada momento. Asumiendo esta idea, los usuarios sólo serán conscientes de los enlaces que denominamos de perspectiva. Por ello, es necesario que todas las posibilidades de navegación que se quieran representar en HDM deben estar traducidos a este tipo de enlaces.

En nuestro ejemplo, tendríamos una única entidad: el bien mueble. Esta entidad , al ser sencilla no tendrá componentes diferentes, pero sí unidades y perspectivas. En principio tendrá dos unidades: una que son los datos de identificación y otra que tendrá los datos de descripción. Estas unidades pueden tener diferentes perspectivas. En el modelo ejemplo que se muestra en la figura 4, hemos resaltado dos perspectivas de la unidad de descripción: una para los arqueólogos, que no incluiría a los autores, ni la fecha ni la certeza, y otra para el resto de los actores. En la figura se muestra además un enlace entre la unidad de identificación y la de descripción. Este es un enlace de perspectiva que nos indica que entre estas unidades se puede navegar. Se han omitido los atributos de las entidades para simplificar el dibujo.

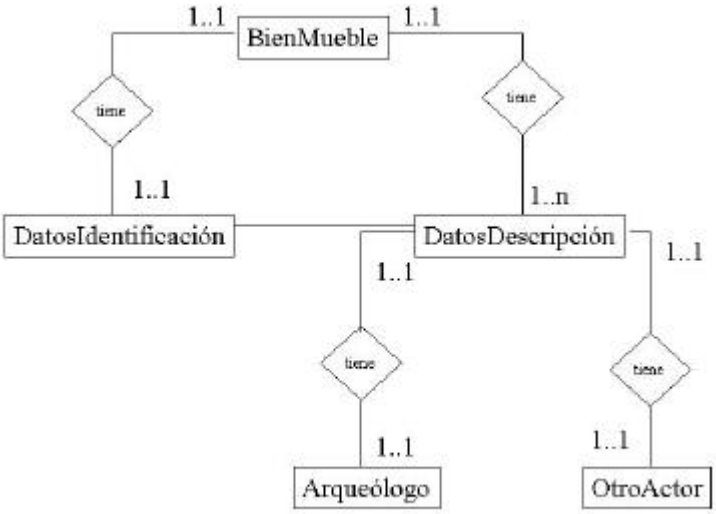

FIGURA 4: EJEMPLO DE MODELO HDM

2.Conclusiones

Como puntos destacables de HDM se puede resaltar el hecho de que es el primer acercamiento a plantear el desarrollo de las aplicaciones multimedia de una manera formal. Fue el pionero en plantear el modelado las aplicaciones multimedia de manera que se estudiaran y diseñaran aspectos tan importantes como la navegación.

En esta técnica se ve la necesidad de separar la información que se almacena, entidad, con la forma en la que se presenta al usuario, perspectiva. Esta idea la vamos a encontrar en la mayoría de las propuestas que veremos, pues resulta muy conveniente a la hora de trabajar con aplicaciones multimedia, y como veremos en el apartado 6, con sistemas de información global.

Sin embargo, HDM no supone una metodología para el desarrollo de aplicaciones multimedia, es simplemente una técnica de modelado. Es cierto que los elementos definidos por HDM (entidades, perspectivas, enlaces, unidades, etc.) sirven para definir este tipo de aplicaciones, pero resultan insuficientes para guiar al diseñador en el proceso de desarrollo de las mismas.

Otro problema esencial que se puede resaltar en HDM es que ha quedado un poco obsoleta, en el sentido de que actualmente las tendencias de diseño están encaminadas hacia el paradigma de la orientación a objetos. En base a este problema, de HDM han surgido nuevas propuestas como EORM u OOHDM, que asumiendo sus conceptos y objetivos, definen una metodología orientada objeto para el diseño de aplicaciones multimedia.

Pero un aspecto que se echa en falta en un modelo de HDM es que no trata los aspectos de interfaz y de múltiples medios de una manera concreta. HDM asume que estos aspectos se tratarán en un nivel más bajo de desarrollo.

En resumen, HDM va a sentar las bases para futuras propuestas de desarrollo, ofreciendo ideas como la separación de lo conceptual, información que se almacena, y de la presentación, información que se presenta. Es además el primer intento de normalizar el desarrollo de aplicaciones multimedia pero queda muy lejos de ser una propuesta metodológica para el desarrollo de sistemas de información global.

5.3.2. RMM- Relationship Management Methogology

RMM es propuesta en 1995 por Tomas Izsakowitz, Arnold Kamis y Marios Kounfaris [Isakowitz 1995]. Se puede considerar una metodología pues asume las etapas de análisis y diseño. RMM propone un proceso basado en 7 fases o etapas en las que el diseñador va modelando la estructura de la aplicación y las posibilidades de navegación de la misma.

La propuesta está basada en el modelo Entidad-Relación y en HDM. Partiendo de ellos define un nuevo modelo el RMDM, que propone un lenguaje que permitirá describir los objetos del dominio, sus interrelaciones y los mecanismos de navegación hipermedia de la aplicación.

1.Conceptos básicos de RMM

RMM asume las extensiones que HDM incluye en los clásicos E-R y añade un nuevo concepto que denomina *slice*. Un slice es un subconjunto de atributos de una entidad que van a ser presentados de forma agrupada al usuario. De esta forma, una aplicación estará formada por entidades cuyos atributos son agrupados en *slices*. En la figura 5 vemos cómo se representan los slices.

Por otro lado, es necesario representar los enlaces. Aunque RMM toma a HDM como base, define sus propios enlaces y los divide en varios grupos:

- Enlaces no condicionales, pueden ser unidireccionales o bidireccionales. Serían los enlaces en los que al partir del origen se pasa al destino sin necesidad de que el usuario indique ninguna condición.
- Enlaces condicionales, en ellos el usuario tiene que indicar alguna condición específica. Dependiendo de dicha condición el usuario irá a un slice u otro.
- Rutas guiadas, se activan automáticamente ante un evento o pasado un tiempo, el usuario no debe hacer nada.

Existe la posibilidad de mezclar varios de estos enlaces. Por ejemplo, podemos optar por varias rutas guiadas dependiendo de las condiciones que introduzca el usuario. En la figura 5 se muestra cómo representarían en el modelo Entidad-Relación estos conceptos.

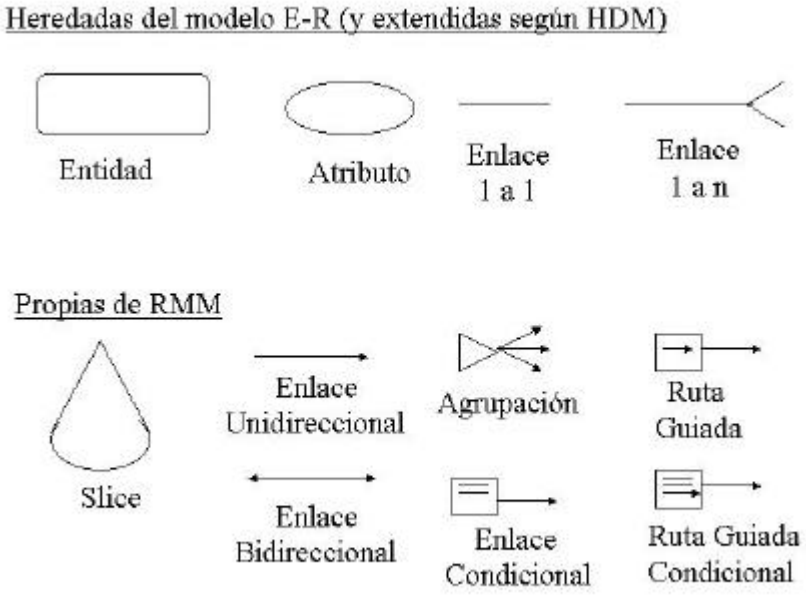

FIGURA 5: PRIMITIVAS DEL MODELO RMDM

Basándose en todas estas ideas, en RMM se representa la aplicación mediante un modelo, denominado RMDM (Relationship Management Data Model). Este modelo es un enriquecimiento del modelo Entidad-Relación y permitirá representar a las aplicaciones multimedia. En este modelo podemos encontrar elementos propios de la propuesta del modelo Entidad-Relación (entidades, atributos, etc.) aunque con las extensiones de HDM y los nuevos conceptos definidos anteriormente (enlaces, rutas guiadas, slice, etc). En la figura 5 se pueden ver todos los iconos que se pueden encontrar en un modelo RMDM.

2.Fases de RMM

Como ya se ha comentado, RMM propone un proceso dividido en etapas para el desarrollo de las aplicaciones multimedia. En este apartado se verán muy brevemente estas fases y sus objetivos.

Fase 1- Realizar el modelo E-R

En esta fase se debe obtener un modelo Entidad-Relación del sistema, sin necesidad de entrar en detalles de navegación o de presentación al usuario. Se debe actuar de la misma forma que se actuaría para obtener un modelo E-R de una aplicación software clásica.

Fase 2- Realizar los diseños de slice

Para cada entidad detectada en la fase anterior, se debe definir un diagrama de slices. Esto es, se deben detectar los slices para esa entidad, es decir, cómo se van a presentar los atributos de la entidad al usuario. Se debe obtener un modelo compuesto por slices y enlaces (ya sean guiados, direccionales, etc), mediante la representación que se mostró en la figura 5.

Así para nuestro ejemplo, habría una entidad que sería el bien mueble, que tendría asociados dos slices: uno para los datos de identificación (código, denominación, acceso, observaciones, ubicación habitual, ubicación temporal) y otro para los datos de descripción (tipologías, estilos, autores, períodos históricos, imágenes, cronología y certeza). Del slice de descripción partiría un enlace unidireccional al slice de identificación, lo que significa que se podrá navegar directamente. Desde el slice de identificación al de descripción lo que nace es un enlace condicional pues el usuario tendrá que indicar si es o no arqueólogo para navegar a dicho slice. En la figura 6 vemos el modelo para nuestro ejemplo. Nuevamente se han omitido los atributos para hacer más sencilla la figura.

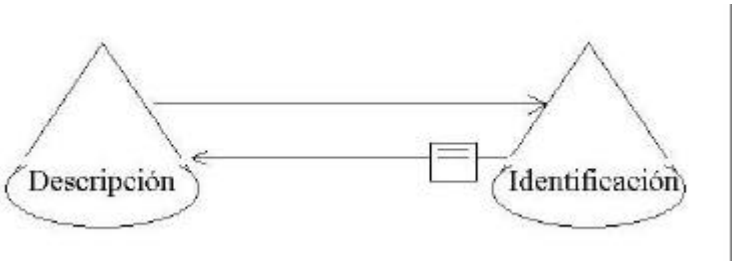

FIGURA 6: DIAGRAMA DE SLICES PARA EL EJEMPLO

Con esta representación, la descripción del sistema no queda claro. Sería necesario acompañarlo por un diccionario de datos que indicara qué contiene cada slice y cuál es la condición del enlace condicional. En principio el formato de este diccionario quedaría a elección del grupo de trabajo.

Fase 3- Diseñar la navegación

Una vez que ya se han definido los slices, se debe diseñar cómo se pasará de una entidad a otra, es decir, hay que enriquecer el modelo Entidad-Relación obtenido en la primera fase con los enlaces entre entidades. En principio en nuestro modelo no sería necesario al tener una única entidad.

El modelo RMDM de la aplicación será la unión del resultado de la fase 3 y de la fase 2.

Fase 4- Definir el protocolo de conversión

En esta fase se debe definir el proceso a seguir para pasar del modelo RMDM a la plataforma de desarrollo concreta. En principio no se propone ninguna técnica estándar a seguir para ello.

Fase 5- Diseñar la interfaz

En esta fase se diseñan las pantallas tal y como se van a mostrar al usuario. Por regla general cada slice se va a corresponder con una pantalla. En esta fase ya es necesario entrar en aspectos concretos del lenguaje de programación que se va a usar.

Fase 6- Implementar la aplicación

En base al protocolo establecido en la fase 4 y al modelo RMDM obtenido, se implementa el sistema.

Fase 7- Probar la aplicación

Una vez que se obtiene la aplicación ejecutable, se deben realizar las pruebas de funcionamiento a la misma. Para ello es necesario definir el test de prueba y estudiar sus resultados. Aunque EORM no ofrece ninguna técnica concreta a seguir para ello.

3.Conclusiones

RMM sube un paso más con respecto a HDM. Propone una metodología basada en el modelo E-R y en HDM para representar las aplicaciones multimedia. Debido a esto es precisamente por lo que no ha tenido demasiada difusión. Es una técnica que se basa en el E-R cuando en su época (1995) todas las tendencias se dirigían a la orientación a objetos.

Algo ventajoso y que hace interesante a RMM es que propone un proceso estructurado y definido a seguir para el desarrollo de estas aplicaciones. En este proceso, sin embargo, se echan en falta las primeras etapas a tener en cuenta en cualquier proceso de desarrollo software, como la captura de requisitos. Además, el proceso que ofrece es demasiado abierto en sus fases como para considerarse como una herramienta de desarrollo adecuada, puesto que en la única fase en la que indica una técnica es en la que se hace uso del modelo RMDM (fase 2). Las otras fases quedan abiertas a la opción del diseñador.

5.3.3. EORM- Enhanced Object Relationship Methodology

1.Conceptos básicos de EORM

Al igual que OOHDM, que veremos más adelante, EORM [Lange 1995] es una de las metodologías de diseño de aplicaciones multimedia más referenciadas en todos los trabajos. Nace igualmente a partir de RMM y HDM pero se orienta ya al paradigma de la orientación a objetos.

EORM propone un proceso iterativo que consiste en enriquecer un modelo de objetos para representar las relaciones existentes entre objetos (*enlaces*). Se estructura en tres fases: análisis, diseño y construcción. Veamos qué objetivos tienen cada una de ellas.

2. Fases de EORM

Fase 1- Análisis

Realmente esta fase no puede denominarse análisis. Se correspondería más a un diseño de objetos, hablando en términos de OMT [Rumbaugh 1995]. Consiste en hacer un modelo orientado a objetos, según las pautas y nomenclatura de OMT para representar la aplicación. La nomenclatura de OMT no vamos a verla en este trabajo. OMT es una metodología de diseño clásico que ha servido como base para UML [Booch 1999] y el Proceso Unificado [Jacobson 1999]. Se ha obviado de este trabajo puesto que todo lo que enunciaba ha quedado obsoleto o ha sido asumido por UML, que es el que actualmente se utiliza. Sin embargo, las propuestas de OMT no sólo han sido tomadas para UML sino también para otras metodologías como EORM u OOHDM, como ya veremos. A pesar de que OMT ha sido la base de EORM y OOHDM, en las últimas versiones de estos se ha asumido la nomenclatura de UML y esta es la que vamos a usar en nuestro ejemplo.

En esta primera etapa de análisis de EORM no se tendrán en cuenta aspectos como la navegación o la interfaz, dejándose ambas para etapas posteriores.

En nuestro ejemplo, tendríamos un modelo de clases bastante sencillo. Lo vemos en la figura 7. Vemos que los datos de tipologías, imagenes, estilos, períodos y autores se modelan como clases que se relacionan con una clase principal BienMueble.

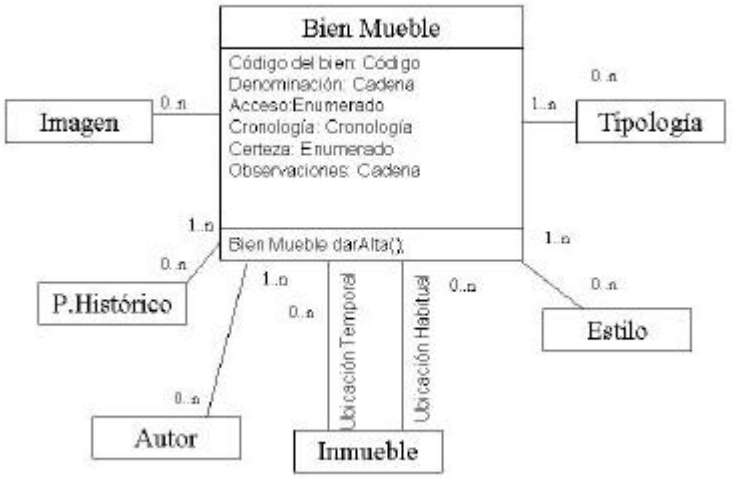

FIGURA 7: EJEMPLO DE MODELO DE ANÁLISIS DE EORM

Fase 2- Diseño

Durante el diseño se procede a modificar el modelo de objetos obtenido en la fase anterior añadiendo semántica suficiente a las relaciones para representar los enlaces. Este modelo de objetos enriquecido se denomina EORM y en él se van a reflejar tanto la estructura de la información (modelo abstracto hipermedial compuesto por nodos y enlaces) como las posibilidades de navegación ofrecidas por el sistema. Para recoger esto último, existirá un repositorio o librería de clases de enlaces, donde se especificarán las posibles operaciones asociadas a cada enlace de un hiperdocumento. Estas operaciones serán del tipo *crear, eliminar, siguiente, etc.* Pero además en estas librerías

se recogerán los atributos del enlace: *fecha de creación, forma de presentación en la pantalla, etc.*

Para estas librerías los autores de EORM dan una serie de tipos de enlaces ya predefinidos. A continuación se describen de forma sencilla.

- simpleLink: es un enlace dentro del mismo documento o la misma página.
- navigationalLink: es un mecanismo que representa un hiperenlace. Esta clase posee una función de backtracking que permite volver a la página origen.
- nodeToNode: es un enlace que une dos clases del modelo.
- spanToNode: este enlace une el contenido de un objeto con otro objeto.
- structureLink: es un conjunto de enlaces simples, por ejemplo, representa un menú.
- setLink: es un structureLink que da acceso a una colección de objetos sin importar el orden.
- listLink: es un structureLink que da acceso a una colección de objetos según un determinado orden.

Así en nuestro ejemplo se definirían dos nodos: uno para los datos de descripción y otro para los datos de identificación. Entre ellos, habrá un enlace del tipo nodeToNode. Además las clases de tipología, autor, imagen, estilos y períodos históricos pasarán a tratarse como nodos y entre ellas y el nodo de descripción, existirán enlaces de tipo setLink, pues referenciará a lista no ordenadas de objetos de estas clases. Vemos cómo quedaría este modelo para el ejemplo en la figura 8.

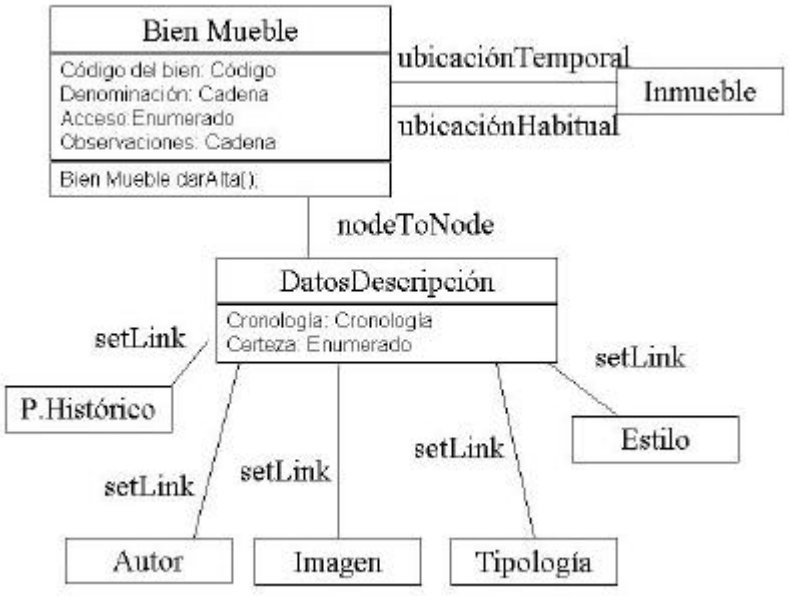

FIGURA 8: EJEMPLO DE MODELO DE DISEÑO DE EORM

Fase 3- Construcción

En esta fase se prepararía el código fuente para cada una de las clases y la interfaz gráfica de usuarios. No se da ninguna recomendación especial para ello.

3. Conclusiones

EORM es, como hemos visto, una metodología sencilla, que asume la orientación a objetos como paradigma para el desarrollo de aplicaciones multimedia. Con ello garantiza todas las ventajas que la orientación a objetos ofrece, pero además aumenta las posibilidades de reutilización en las aplicaciones, gracias a la definición del repositorio o librerías de clases enlace.

EORM también es adecuada porque, siguiendo la idea inicial de HDM, separa la navegación de lo conceptual. Esto garantiza la reutilización y un mantenimiento más fácil. Si hay un cambio en la navegación, lo conceptual no se modifica.

Por otro lado, la aplicación de la metodología EORM puede resultar bastante sencilla gracias al uso de ODMTool, una herramienta desarrollada por el propio autor de EORM. Esta herramienta se usa de forma conjunta con el generador comercial de interfaces gráficas de usuario, ONTOS Studio y con un sistema gestor de base de datos orientado a objetos, de manera que se permite el diseño interactivo de esquemas EORM y la generación automática de código, inicialmente en C++.

A pesar de todas estas ventajas, a EORM se le pueden hacer ciertas críticas. Por un lado, el proceso metodológico que propone resulta insuficiente en muchos casos principalmente porque solo trata de manera específica los aspectos de almacenamiento y navegación, dejando a un lado temas como la funcionalidad del sistema o los aspectos de interfaz. Además, en ningún momento comenta las técnicas a seguir para obtener los modelos que propone o los productos que se deben generar en el desarrollo.

EORM también deja a un lado un aspecto muy importante en la mayoría de las aplicaciones: la captura de requisitos. No sólo no ofrece ninguna propuesta sino que no indica ninguna que se pueda usar.

En conclusión, EORM es la primera propuesta orientada a objetos. Es necesario tenerla en cuenta a la hora de realizar una propuesta por las ideas que ofrece y por los modelos que plantea, que resultan muy adecuados para representar la navegación y lo conceptual. Pero deja a un lado aspectos que son críticos en el ciclo de desarrollo.

5.3.4.The MacWeb Hypermedia Development Enviroment

1. Conceptos básicos de *MacWeb Hypermedia Developement Enviroment*

El entorno de desarrollo hipermedia, MacWeb, propuesto por los hermanos Nanard en 1995 [Nanard 1995] no se puede considerar una propuesta metodológica propiamente dicha. En el entorno MacWeb, una aplicación hipermedia se desarrolla en base a la interfaz. Los autores de este entorno puntualizan en el hecho de que lo más importante de una aplicación web es la comunicación con el usuario. Por ello, centran todo el proceso de desarrollo en la interfaz.

Como veremos en el siguiente apartado, el diseño de una aplicación hipermedia debe dividirse en dos grandes fases: el desarrollo del proceso mental y la realización de los pasos metodológicos.

2. Fases de MacWeb

El proceso mental incluye los siguientes pasos:

- Generación del material, consiste en conseguir toda la información, ya sea textual, gráfica, o de cualquier tipo, que va a mostrarse en la aplicación.
- Organizar y estructurar este material.
- Reorganizar el material y actualizarlo cuando sea necesario.
- Evaluación del material y de la estructuración realizada.

El proceso de desarrollo engloba los pasos clásicos, como son:

- Desarrollo del modelo conceptual.
- Desarrollo del modelo navegacional.
- Desarrollo de la interfaz abstracta
- Implementación.
- Pruebas

El orden en que se realizan estas fases depende de lo que el diseñador desee. En principio las tareas mentales y las de desarrollo se realizan en paralelo, aunque bien se podrían realizar las mentales y luego las de desarrollo.

3. Conclusiones

Esta propuesta no ofrece ninguna nueva aportación en cuanto a técnicas o modelos, por lo que no vamos a aplicarla a nuestro ejemplo. El entorno de desarrollo de MacWeb propone el uso de técnicas clásicas de la orientación a objetos, como son la generalización y la instanciación para representar los aspectos de navegación. Junto con MacWeb, se construyó una herramienta que ayuda a realizar el proceso de desarrollo. Esta herramienta posee clases predefinidas como son los nodos o los enlaces.

El entorno de desarrollo de MacWeb resulta interesante por la separación que propone entre los aspectos de contenido y desarrollo. Esta separación aparece en los desarrollos reales en la que los especialistas informáticos tienen que desarrollar una aplicación multimedia sobre un tema del que, en la mayoría de los casos no son expertos. Cada vez es más necesaria la colaboración entre los especialistas temáticos y los técnicos, por lo que es necesario tener en cuenta en el proceso de desarrollo ambos aspectos. Además es la primera propuesta que tiene en cuenta al usuario en el ciclo de desarrollo del sistema.

5.3.5. OOHDM- Object-Oriented Hypermedia Design Method

OOHDM es una metodología de desarrollo propuesta por Rossi y Schwabe [Rossi 1996] para la elaboración de aplicaciones multimedia. OOHDM está basada en HDM, en el sentido de que toma muchas de las definiciones, sobre todo en los aspectos de navegación, planteadas en el modelo de HDM. Sin embargo, OOHDM supera con creces a su antecesor, ya que no es simplemente un lenguaje de modelado, sino que define unas pautas de trabajo, centrado principalmente en el diseño, para desarrollar aplicaciones multimedia de forma metodológica.

OOHDM ha evolucionado bastante desde su nacimiento. Actualmente está siendo utilizado por sus autores para el desarrollo de aplicaciones en la web [Schwabe 2001].

1. Conceptos básicos de OOHDM

OOHDM como ya se ha comentado es una metodología de desarrollo para aplicaciones multimedia. Antes de comenzar a detallar cada una de las fases que propone, es necesario resaltar algunas de sus características.

La primera de ellas es que OOHDM está basada en el paradigma de la orientación a objetos. En esto se diferencia de su antecesor HDM. A la hora de la representación de las clases y los diagramas, OOHDM utiliza la nomenclatura propuesta por OMT [Rumbaugh 1995], como ya se comentó al estudiar EORM. Sin embargo en las últimas versiones ya se ha asumido UML.

Otra característica de OOHDM es que, a diferencia de HDM, no sólo propone un modelo para representar a las aplicaciones multimedia, sino que propone un proceso predeterminado para el que indica las actividades a realizar y los productos que se deben obtener en cada fase del desarrollo.

Sin embargo, OOHDM no puede considerarse una metodología en el amplio sentido, ya que, aunque se detalla el proceso a seguir en lo que sería el diseño de la aplicación, no toma parte en otras fases como pueden ser la captura de requisitos o el análisis. Según OOHDM estas fases son idénticas a las que se deben realizar en la propuesta de OMT.

Fundamentalmente OOHDM toma como partida el modelo de clases que se obtiene en la fase de diseño de objetos de OMT o, en sus últimas versiones, en el análisis del Proceso Unificado de UML. A este modelo lo denomina *modelo conceptual*. El proceso anterior a éste, es decir, el análisis y el diseño de sistemas, sería idéntico al que se realizaría en el desarrollo de un sistema clásico según OMT o UML.

Partiendo de este modelo conceptual, OOHDM propone ir añadiendo características que permitan incorporar a esta representación del sistema todos los aspectos propios de las aplicaciones multimedia. En una segunda etapa de diseño, se parte de ese modelo conceptual y se añade a éste todos los aspectos de navegación, obteniéndose un nuevo modelo de clases denominado *modelo navegacional*. Por último, este modelo sirve como base para definir lo que en el argot de OOHDM se denomina *modelo de interfaz abstracta.* El modelo de interfaz abstracta representa la visión que del sistema tendrá cada usuario del mismo.

A continuación vamos a ver un pequeño resumen de las fases de OOHDM, de sus objetivos y de los productos que resultan al aplicar cada una de ellas.

2. Fases de OOHDM

En OOHDM se proponen 4 fases de desarrollo:

- Diseño Conceptual
- Diseño Navegacional
- Diseño de Interfaz Abstracto
- Implementación

OOHDM es una mezcla de estilos de desarrollo basado en prototipos, en desarrollo interactivo y de desarrollo incremental. En cada fase se elabora un modelo que recoge los aspectos que se trabajan en esa fase. Este modelo parte del modelo conseguido en la fase anterior y sirve como base para el modelo de la siguiente fase.

Fase 1- Diseño Conceptual

Se construye un modelo orientado a objetos que represente el dominio de la aplicación usando las técnicas propias de la orientación a objetos. La finalidad principal durante esta fase es capturar el dominio semántico de la aplicación en la medida de lo posible, teniendo en cuenta el papel de los usuarios y las tareas que desarrollan. El resultado de esta fase es un modelo de clases relacionadas que se divide en subsistemas. En la tabla 2 se esquematiza esta fase.

Fase	Diseño conceptual
Productos	Diagrama de Clases, División en subsistemas y relaciones
Herramientas	Técnicas de modelado O.O, patrones de diseño
Mecanismos	Clasificación, agregación, generalización y especialización
Objetivo de diseño	Modelo semántico de la aplicación

TABLA 2: FASE DE DISEÑO CONCEPTUAL DE OOHDM

Debido a la similitud de OOHDM y EORM, y debido a que ambos sistemas parten del modelo de clases básico del sistema, el resultado de esta primera fase de OOHDM para nuestro ejemplo sería igual que el de EORM y que vimos en la figura 7.

Fase 2- Diseño Navegacional

En OOHDM una aplicación se ve a través de un sistema de navegación. En la fase de diseño navegacional se debe diseñar la aplicación teniendo en cuenta las tareas que el usuario va a realizar sobre el sistema. Para ello, hay que partir del esquema conceptual desarrollado en la fase anterior. Hay que tener en cuenta que sobre un mismo esquema conceptual se pueden desarrollar diferentes modelos navegacionales (cada uno de los cuales dará origen a una aplicación diferente).

La estructura de navegación de una aplicación hipermedia está definida por un esquema de clases de navegación específica, que refleja una posible vista elegida. En OOHDM hay una serie de clases especiales predefinidas, que se conocen como clases navegacionales: *Nodos, Enlaces y Estructuras de acceso*, que se organizan dentro de un *Contexto Navegacional*. Mientras que la semántica de los nodos y los enlaces son comunes a todas las aplicaciones hipermedia, las estructuras de acceso representan diferentes modos de acceso a esos nodos y enlaces de forma específica en cada aplicación.

1- Nodos: Los nodos son contenedores básicos de información de las aplicaciones hipermedia. Se definen como vistas orientadas a objeto de las clases definidas durante el diseño conceptual usando un lenguaje predefinido y muy intuitivo, permitiendo así que un nodo sea definido mediante la combinación de atributos de clases diferentes relacionadas en el modelo de diseño conceptual. Los nodos contendrán atributos de tipos básicos (donde se pueden encontrar tipos como imágenes o sonidos) y enlaces.

2- Enlaces: Los enlaces reflejan la relación de navegación que puede explorar el usuario. Ya sabemos que para un mismo esquema conceptual puede haber diferentes

esquemas navegacionales y los enlaces van a ser imprescindibles para poder crear esas vistas diferentes.

OOHDM hace una definición de clase enlace que contiene un único atributo. Lo vemos en la figura 9. Su atributo almacenaría la clase a la que se navega por ese enlace. Las clases enlaces sirven para especificar los atributos de enlaces y estos a su vez para representar enlaces entre clases nodos o incluso entre otros enlaces. En cualquier caso, el enlace puede actuar como un objeto intermedio en un proceso de navegación o como un puente de conexión entre dos nodos.

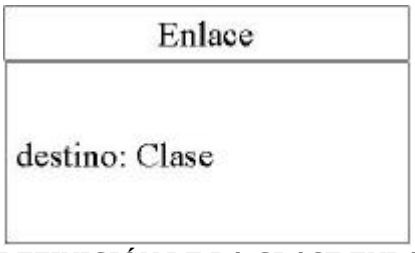

FIGURA 9: DEFINICIÓN DE LA CLASE ENLACE

3- Estructuras de Acceso: Las estructuras de acceso actúan como índices o diccionarios que permiten al usuario encontrar de forma rápida y eficiente la información deseada. Los menús, los índices o las guías de ruta son ejemplos de estas estructuras. Las estructuras de acceso también se modelan como clases, compuestas por un conjunto de referencias a objetos que son accesibles desde ella y una serie de criterios de clasificación de las mismas.

4- Contexto Navegacional: Para diseñar bien una aplicación hipermedia, hay que prever los caminos que el usuario puede seguir, así es como únicamente podremos evitar información redundante o que el usuario se pierda en la navegación. En OOHDM un contexto navegacional está compuesto por un conjunto de nodos, de enlaces, de clases de contexto y de otros contextos navegacionales. Estos son introducidos desde clases de navegación (enlaces, nodos o estructuras de acceso), pudiendo ser definidas por extensión o de forma implícita.

5- Clase de Contexto: Es otra clase especial que sirve para complementar la definición de una clase de navegación. Por ejemplo, sirve para indicar qué información está accesible desde un enlace y desde dónde se puede llegar a él.

En la figura 10 encontramos el diagrama de clases navegacionales para nuestro ejemplo. En ella aparecen dos nodos: DatosIdentficiación y DatosDescripción. El primero muestra los datos de identificación del bien, que se describieron en el apartado 2 y en el segundo los datos de descripción.

Nótese que han aparecido dos atributos nuevos de tipo Enlace en estas clases. Son enlaces que indican que podemos navegar a la clase MenúMuebles desde estos nodos. Por su parte, esta clase es una estructura de acceso. Vemos entonces como en los modelos navegacionales aparecen instancias de la clase enlace para representar los enlaces entre nodos.

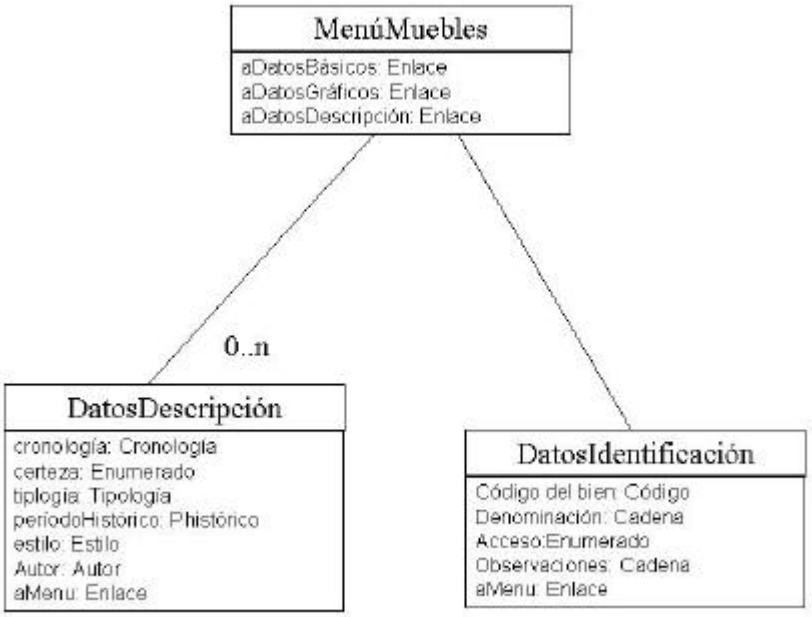

FIGURA 10: EJEMPLO DE DIAGRAMA DE CLASES NAVEGACIONALES

Este modelo quedaría demasiado ambiguo por sí solo. Es necesario indicar de qué clase o clases del modelo conceptual proviene cada una y cuáles son los atributos de la clase enlace. Por ello, este modelo de clases iría acompañado de una descripción formal de lo que son. En la figura 11 vemos la descripción formal de dos de estas clases como indica OOHDM.

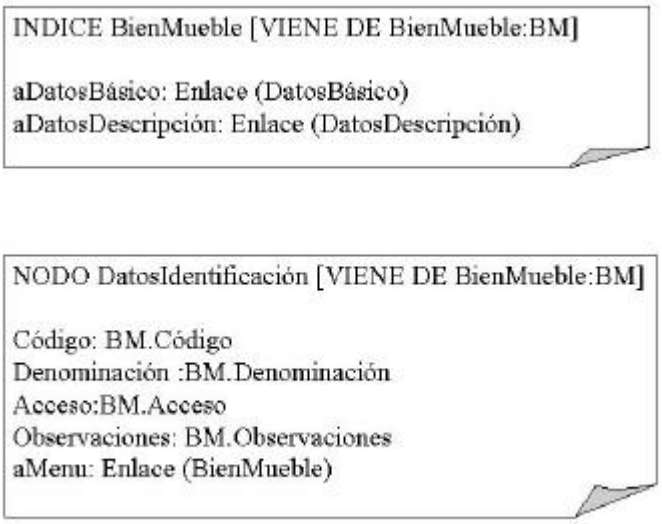

FIGURA 11: EJEMPLO DE DESCRIPCIÓN DE CLASES NAVEGACIONALES

En el primer ejemplo se describe la clase índice que tenemos, indicando que toma datos de la clase BienMueble del diagrama de clases conceptual. Se describen sus dos atributos como enlaces y se indica a qué clase navegarían. En el segundo, se describe uno de los nodos. Aquí se indica que toma los atributos de la clase BienMueble del modelo conceptual y se hace una equivalencia al atributo de esta clase asumido. Por ejemplo, el Código del nodo es igual al atributo Código de la clase BienMueble del modelo conceptual.

La navegación no se encontraría definida sin el otro modelo que propone OOHDM: el contexto navegacional. Esto es la estructura de la presentación dentro de un determinado contexto. Los contextos navegacionales son uno de los puntos más criticados a OOHDM debido a su complejidad de expresión. Se va a plantear un contexto de presentación sencillo para nuestro sistema de ejemplo. En la figura 12 vemos como quedaría.

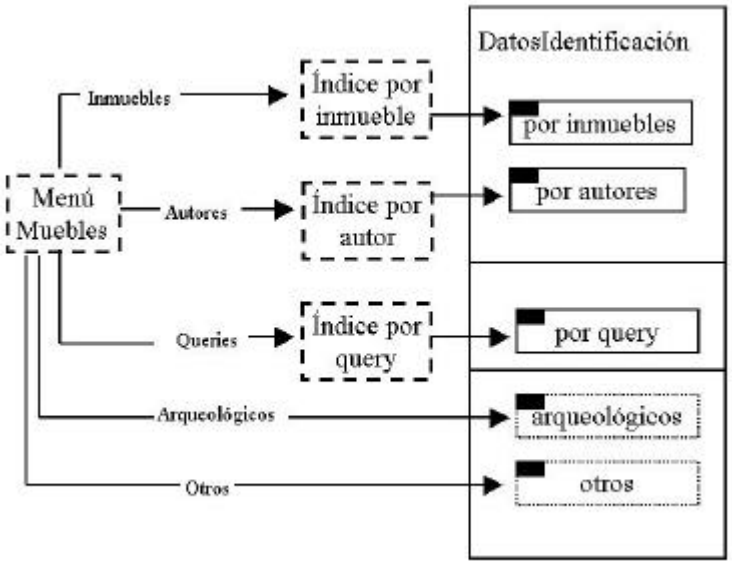

FIGURA 12: EJEMPLO DE CONTEXTO NAVEGACIONAL

Con este contexto se está expresando lo siguiente. Hay una clase principal denominada MenuMuebles, que es la clase índice del modelo navegacional, y que es la partida del contexto. En este menú aparecen 3 índices que nos permiten consultar los bienes a través de los inmuebles, es decir, detallar todos los bienes muebles que pertenecen aun determinado inmueble; a través de los autores, detallando los bienes muebles de un determinado autor; y a través de queries o sentencias de búsqueda libres, indicamos el criterio de búsqueda y obtenemos los bienes muebles que la cumplen. Ejecutando cualquiera de estas opciones llegaríamos a ver los datos de identificación ordenadas por inmuebles, por autores o por query de los bienes muebles. Como puede verse, cuando el usuario tiene posibilidad de ejecutar una sentencia de búsqueda se indica con un cuadro punteado en negrita en el que se indica el concepto por el que se puede buscar. Cada criterio de búsqueda genera una vista diferente de los muebles que se representa mediante cajas con el borde negro continuo. Estas cajas tienen en su borde superior izquierdo un recuadro negro que indica que hay un orden predefinido para mostrarlo.

Hay casos en los que hay un grupo de elementos caracterizados por algo. Por ejemplo, los bienes muebles arqueológicos tienen un interés especial para los arqueólogos. En este caso, vemos que desde el menú nace una flecha que va a los datos de identificación hacia una caja punteada denominada “arqueológicos”. Esto indica que desde el menú principal el usuario tendría la opción de ver solo los bienes arqueológicos. Aquí no es necesaria hacer ninguna selección antes de obtener los bienes, por eso no caja índice intermedia. La opción de “otros” indicaría que se pueden obtener los bienes no arqueológicos.

Nótese que con sólo dos criterios de búsquedas y trabajando con una sola clase navegacional, la de datos de identificación, el diagrama resulta bastante complejo de

explicar. Pero además resulta algo ambiguo porque no podemos indicar por qué atributos se pueden hacer las query.

Cuando los criterios de recuperación se complican, caso muy común en la mayoría de las aplicaciones, el contexto navegacional resulta bastante engorroso por la cantidad de opciones que saldrían. Imaginemos que queremos buscar por: autor, tipología, estilo, período histórico, ubicación actual y por cualquier combinación de ellos. La cantidad de cajas de índice que saldrían resultarían inviables a la hora de la representación.

TRANSFORMACIÓN NAVEGACIONAL: cuando la aplicación se ejecuta en una sola vista, el contexto navegacional tiene bastante poder semántico como para representar la aplicación, por muy compleja que pueda salir. Sin embargo, cuando pueden aparecer diferentes contextos de navegación a la vez, se requiere el uso de transformaciones navegacionales. A través de ellas podemos expresar concurrencias o navegaciones que se ejecutan a la par. Por ejemplo, imaginemos que nuestra aplicación variara en su representación para los arqueólogos y el resto de investigadores, pues habría que representar diferentes contextos navegacionales.

En la tabla 3 vemos un resumen de esta fase.

Fase	Diseño navegacional
Productos	Nodos, enlaces, estructuras de accesos, contextos navegacionales y transformaciones navegacionales
Herramientas	Técnicas de modelado O.O, patrones de diseño, diagramas de estados, escenarios
Mecanismos	Clasificación, agregación, generalización y especialización
Objetivo de diseño	Establecer los recorridos que el usuario puede seguir por la aplicación

TABLA 3: FASE DE DISEÑO NAVEGACIONAL DE OOHDM

Fase 3- Diseño de Interfaz Abstracta

Una vez definida la estructura navegacional, hay que prepararla para que sea perceptible por el usuario y esto es lo que se intenta en esta fase. Esto consiste en definir qué objetos de interfaz va a percibir el usuario, y en particular el camino en el cuál aparecerán los diferentes objetos de navegación, qué objeto de interfaz actuarán en la navegación, la forma de sincronización de los objetos multimedia y el interfaz de transformaciones. Al haber una clara separación entre la fase anterior y esta fase, para un mismo modelo de navegación se pueden definir diferentes modelos de interfaces, permitiendo, así que el interfaz se ajuste mejor a las necesidades del usuario.

MODELOS DE VISTAS ABSTRACTAS DE DATOS (ADVs): los modelos de los ADVs no son más que representaciones formales que se usan para mostrar:

i. La forma en que se estructura la interfaz, para ello se usan las vistas abstractas de datos. Estos son elementos que tienen una forma y un dinamismo. Son elementos abstractos en el sentido de que solo representan la interfaz y su dinamismo, y no la implementación, no entran en aspectos concretos como el color de la pantalla o la ubicación en ésta de la información. Así, tendremos un conjunto de representaciones gráficas, que gestionan las estructuras de datos y de control, y un conjunto de aspectos de interfaz, como las entradas del usuario y las salidas que se le ofrecen.

ii. La forma en que la interfaz se relaciona con las clases navegacionales, para ello se usan diagramas de configuración. Los diagramas de configuración van a ser grafos

dirigidos que permitirán indicar de qué objetos de navegación toman la información los ADV.

iii. La forma en que la aplicación reacciona a eventos externos, para ello se usan los ADVs-Charts. Los ADVs-Charts van a ser diagramas bastante similares a las máquinas de estados, es más en las últimas versiones de OOHDM se usan máquinas de esto. A través de ellas se puede indicar los eventos que afectan a una ADV y cómo ésta reacciona a ese elemento.

Aplicando esta fase a nuestro ejemplo tendríamos varios ADVs, por ejemplo en la figura 13 mostramos el de la clase Menú y el de la clase DatosBásicos. Nótese como se representan los enlaces entre una clase y otra. Aunque aquí hemos optado por representar las clases navegacionales DatosIdentificación y MenúMuebles con un solo ADV cada uno, esto no tiene porqué ser así. Una clase puede dar origen a uno o más ADV y un ADV puede representar a uno o más clases.

FIGURA 13: EJEMPLO DE ADV

Para indicar de qué clase navegacional viene un ADV se usa el diagrama de configuración. Como ya se ha dicho, es un grafo en el que las fuentes son clases ADVs y los sumideros clase navegacionales. En la figura 14 vemos parte de este diagrama para el ejemplo. Con él indicamos que el ADVDatosIdentificación descrito proviene de la clase DatosIdentificación.

FIGURA 14: EJEMPLO DE DIAGRAMA DE CONFIGURACIÓN

Ahora para completar el ejemplo habría que crear la máquina de estados que representa el dinamismo de cada uno de los ADV. No vamos a recogerlo puesto que el ejemplo es muy sencillo y no posee un dinamismo que justifique esto.

Como resumen, en la tabla 4 vemos descrita esta fase.

Fase	Diseño de interfaz abstracta
Productos	Objetos de interfaz abstracta, respuestas a eventos externos y transformaciones de interfaz
Herramientas	ADVs, Diagramas de configuración, ADV-Charts y patrones de diseño
Mecanismos	Mapeado entre los objetos de navegación y los objetos visibles
Objetivo de diseño	Modelado de los objetos perceptibles por el usuario y de cómo le afecta a la aplicación los eventos externos

TABLA 4: FASE DE DISEÑO DE INTERFAZ ABSTRACTO DE OOHDM

Fase 4- Implementación

Una vez obtenido el modelo conceptual, el modelo de navegación y el modelo de interfaz abstracta, sólo queda llevar los objetos a un lenguaje concreto de programación, para obtener así la implementación ejecutable de la aplicación. En la tabla 5 vemos un resumen de esta fase.

Fase	Implementación
Productos	Aplicación ejecutable
Herramientas	El entorno del lenguaje de programación
Mecanismos	Los ofrecidos por el lenguaje
Objetivo de diseño	Obtener la aplicación ejecutable

TABLA 5: FASE DE IMPLEMENTACIÓN DE OOHDM

Para terminar, podríamos decir que los puntos claves de OOHDM se encuentran en:

- Contempla los objetos que representan la navegación como vistas de los objetos detallados en el modelo conceptual.
- Abstrae los conceptos básicos de la navegación: nodos, enlaces e índices y los organiza mediante el uso de los contextos de navegación, permitiendo así una organización adecuada de los mismos.
- Separa las características de interfaz de las características de navegación, con las ventajas que esto supone.

3. Conclusiones

OOHDM es sin duda una de las metodologías que más aceptación ha tenido, y sigue teniendo, en el desarrollo de aplicaciones multimedia. Actualmente está sirviendo como base para el desarrollo de nuevas propuestas metodológicas para los sistemas de información web [Mandel 1999].

OOHDM es una propuesta basada en el diseño, que ofrece una serie de ideas que han sido asumidas por bastantes propuestas y que han dado muy buenos resultados. La primera de ellas es que hace una separación clara entre lo conceptual, lo navegacional y lo visual. Esta independencia hace que el mantenimiento de la aplicación sea mucho más sencillo. Además, es la primera propuesta que hace un estudio profundo de los aspectos de interfaz, esencial no solo en las aplicaciones multimedia, sino que es un punto crítico en cualquiera de los sistemas que se desarrollan actualmente.

OOHDM hace uso también de la orientación a objetos y de un diagrama tan estandarizado como el de clases, para representar el aspecto de la navegación a través de las clases navegacionales: índices, enlaces y nodos. Esta idea ha dado muy buenos resultados y parece muy adecuada a la hora de trabajar.

Sin embargo, y a pesar de esto, OOHDM presenta algunas deficiencias. OOHDM ha dejado fuera de su ámbito un aspecto esencial que es el tratamiento de la funcionalidad del sistema. El qué se puede hacer en el sistema y en qué momento de la navegación o de la interfaz se puede hacer, es algo que no trata y que lo deja como tarea de implementación.

Además, OOHDM no ofrece ningún mecanismo para trabajar con múltiples actores. Por ejemplo, imaginemos que la interfaz y la navegación de la aplicación varia sustancialmente dependiendo de quién se conecte a la aplicación. El diagrama navegacional, los contextos navegacionales y los ADVs resultarían muy complejos para respresentar esta variabilidad.

Otra propuesta de OOHDM que no parece adecuada es la de los contextos navegacionales. Hemos visto que para nuestro sencillo ejemplo, el contexto navegacional es complejo y ambiguo. Cuando el ejemplo se complica el resultado puede llegar a ser ilegible.

En resumen, OOHDM nos ofrece una serie de ideas muy adecuadas a la hora de plantear una metodología de desarrollo que tenga en cuenta la navegación y la interfaz. Existen muchos ejemplos desarrollados mediante OOHDM que pueden ayudarnos a ver la viabilidad de estas ideas. Sin embargo, hay que limitar el uso de esta propuesta a aplicaciones multimedia o de la web sencillas en las que la complejidad funcional sea mínima.

5.3.6. WSDM- Web Site Design Method

1. Conceptos básicos de WSDM

El *Método para diseño de sitios web* (WSDM) es una propuesta en la que el sistema se define en base a los grupos de usuarios. Fue propuesto por De Troyer y Leune en 1997 [De Troyer 1997] y se compone de cuatro fases.

Los autores de WSDM dividen los sitios web en dos grupos: *Kiosco web* y *Aplicación Web* [Isakowitz 1995]. Las que pertenecen al primer tipo ofrecen al usuario una determinada información y les permite navegar hacia ella. Las aplicaciones web englobarían a aquellos sistemas de información interactivos cuya interfaz de usuario es un conjunto de páginas web. Tras esta división, los autores de WSDM especifican que su propuesta es una propuesta orientada a la primera de ellas, no resultando adecuada para las aplicaciones web.

2. Fases de WSDM

El modelo de diseño de sitios web se divide en cuatro fases: modelo de usuario, diseño conceptual, diseño de la implementación e implementación. A su vez, el modelo de usuario se divide en dos subfases: clasificación y descripción. Por su parte el diseño conceptual se divide en otras dos subfases: modelado de objetos y diseño navegacional. En la figura 15 vemos la estructura del modelo.

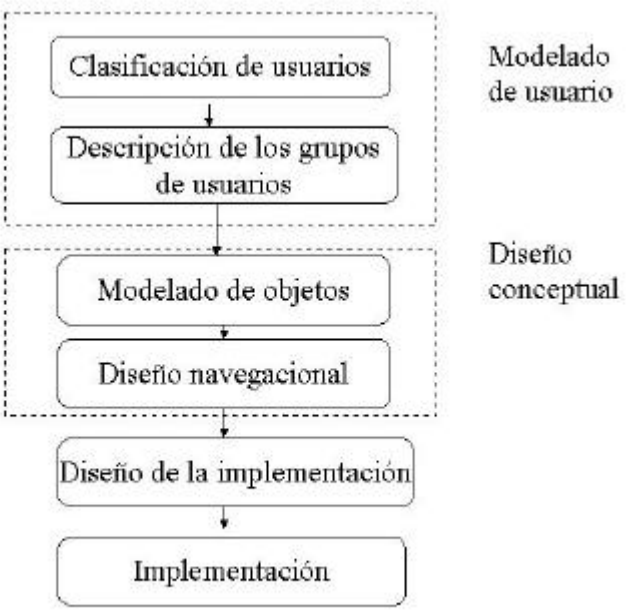

FIGURA 15: ESQUEMA DE FASES DE WSDM

Fase 1- Modelado de usuarios

En esta primera fase se intenta detectar los perfiles de usuarios que se van a presentar en la aplicación. Para ello, se deben realizar dos tareas:

?? Clasificación de usuarios: en este paso se deben identificar y clasificar a los usuarios que va a tener el sistema. Para ello, los autores de WSDM proponen que se estudie el entorno de la organización donde se vaya a implantar el sistema y los procesos que se vayan a generar. De esta forma, se debe generar un diagrama similar al del diagrama de contexto.

?? Descripción de los grupos de usuarios: en esta segunda etapa se describen con más detalles a los grupos de usuarios detectados en la etapa anterior. Para ello, se debe elaborar un diccionario de datos, en principio con formato libre, en el que se indique los requisitos de almacenamiento de información para cada grupo de usuarios y sus características.

En nuestro ejemplo, esta fase daría como resultado la descripción de los grupos de actores que definimos en el apartado 5.2. Como no se propone ningún formato, quedémonos con la descripción informal que se hizo allí.

Fase 2- Diseño Conceptual

La fase de diseño conceptual puede recordar bastante a la metodología OOHDM, vista anteriormente. Sin embargo, vamos a detectar algunas diferencias. La principal es que en WSDM se hace una diferenciación de los grupos de usuarios desde el comienzo, mientras que en OOHDM se espera hasta las últimas fases del diseño para hablar de los mismos. Sin embargo, las dos metodologías proponen realizar primero un modelado conceptual y luego un modelado de navegación.

Así, esta fase en WSDM se divide en las siguientes dos etapas:

?? Modelado de objetos

El objetivo de esta subfase es modelar formalmente los requisitos de información expresados en la etapa de descripción de la fase anterior. Para ello, se debe obtener un modelo de clases para cada grupo de usuarios, a cada modelo se le denominara diagrama de objetos de usuarios. Al conjunto de todos los diagramas de todos los usuarios se le denomina modelo de objetos de usuarios. La nomenclatura de clases se representa mediante la propuesta de OMT (en las últimas versiones ya se asume la de UML). Para nuestro ejemplo, el modelo de objetos sería similar al de clases que vimos en OOHDM y EORM, recogido en la figura 7.

?? Diseño navegacional

Esta fase tiene como objetivo el conseguir un modelo para representar las posibilidades de navegación del sistema. El modelo propuesto por WSDM es bastante sencillo y se basa en representar la navegación a través de un conjunto de pistas de navegación. Una pista de navegación expresa cómo un usuario concreto puede navegar hacia una determinada información.

El modelo es representado mediante un diagrama en el que pueden aparecer 4 elementos que se muestran en la figura 16.

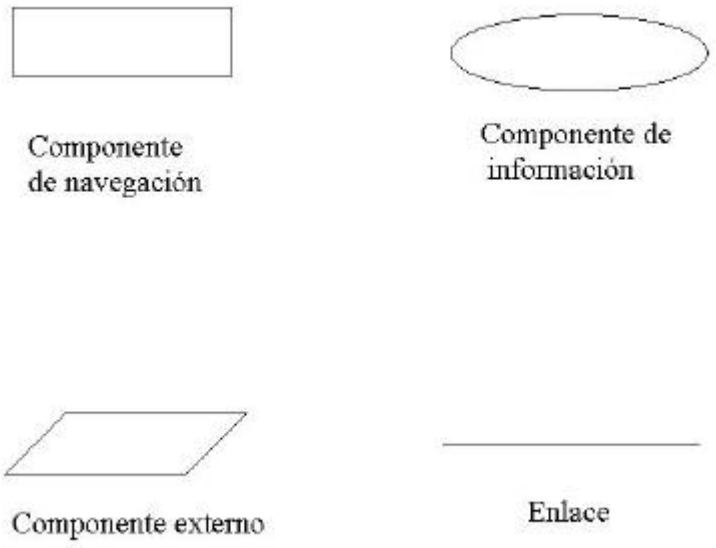

FIGURA 16: REPRESENTACIÓN GRÁFICA DE LOS CONCEPTOS DE NAVEGACIÓN

Cada componente de información se corresponde con una perspectiva de los tipos de objetos (POT) descritos en la subfase anterior. Es decir, una visión particular de la información. Los componentes de información pueden estar relacionados entre ellos a través de enlaces. Los componentes de navegación son conjuntos de enlaces y las entidades externas representan una referencia hacia un elemento externo al sistema.

En WSDM se propone un algoritmo para construir el modelo navegacional a partir del modelo de clases para un determinado usuario, cada uno de estos modelos es un contexto. Es interesante ver este algoritmo pues es el único proceso totalmente

estructurado que se da en las metodologías vistas hasta ahora para conseguir representar la navegación.

Para cada contexto de usuario hay que realizar una serie de pasos que vemos en la figura 17.

1- Detectar el contexto: lo primero que hay que estudiar es cómo empieza el contexto. Esto es, hay que definir cuál es la componente de navegación de partida.

2- Estudiar la información: tras definir el comienzo, se define qué información hay que mostrar y se estructura ésta en una o varias componentes de navegación. Cada componente de navegación se hará corresponder con una página.

3- Estudiar la navegación: Es necesario revisar la navegación definida en la fase anterior. Por regla general, cada perspectiva debe tener un enlace al componente de navegación principal. También se debe asegurar que haya enlaces a todas las entidades externas y que los componentes de navegación definidos en la tarea 2 estén conectadas mediante enlaces de la forma adecuada.

FIGURA 17: PROCESO PARA OBTENER EL MODELO DE NAVEGACIÓN

Aplicándolo a nuestro ejemplo, tendríamos un componente de navegación para los datos de identificación y otro para los datos de descripción, enlazados entre sí, tal y como se ve en la figura 18. Como el primero toma datos de la clase Bien Mueble e Inmueble tiene dos componente de información. El segundo tiene más, puesto que toma datos de más clases.

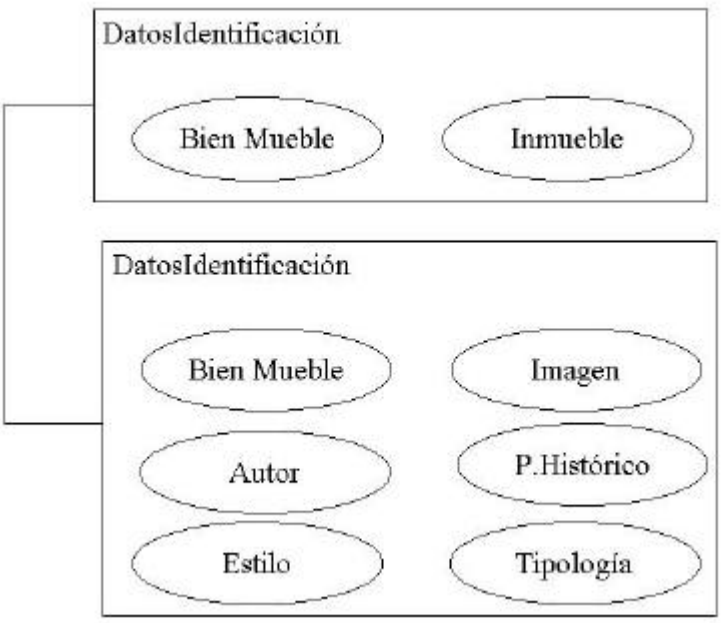

FIGURA 18: EJEMPLO DE DISEÑO NAVEGACIONAL DE WSDM

Fase 3- Diseño de la implementación

Una vez definido el modelo es necesario diseñar la interfaz y el entorno de usuario del sistema. Para ello, los autores de WSDM no proponen nada. Únicamente referencian a las propuestas de December y Ginsberg [December 1995] para el desarrollo de interfaces.

Fase 4- Implementación

En esta fase se pretende conseguir la aplicación ejecutable final en base a los resultados de las fases anteriores. De nuevo tampoco se propone ninguna técnica en concreto para ello.

3. Conclusiones

En principio la propuesta de WSDM puede ser bastante interesante en el sentido de que ofrece una nueva visión al tratamiento de los usuarios. Es una metodología totalmente orientada a diseñar la aplicación en base a los grupos de usuarios desde el principio. Sin embargo, no comenta nada sobre aspectos que pueden resultar importante partiendo de esta idea. Por ejemplo, puede ser que la misma información se presente a dos usuarios de forma diferente. En muchos casos puede resultar complejo para el implementador detectar que se trata de la misma información.

Además, como sus propios autores indican, solo sirve para desarrollar kioscos web, es decir, aplicaciones que muestren información. Esta propuesta no trabaja aspectos como la funcionalidad, la seguridad, etc. necesarias en las aplicaciones web. Se centra solo en cómo presentar la información al usuario.

5.3.7. OO-Method y OO-Hmethod

1. Conceptos básicos de OO-Method

OO-Method [Pastor 1997] es un método orientado a objetos desarrollado en la Universidad Politécnica de Valencia que fusiona el uso de un lenguaje de especificación formal (OASIS [Pastor 1992]) con una notación gráfica tomada de los estándares más usados. Esta aproximación aprovecha las buenas propiedades de los lenguajes de especificación formales con la experiencia acumulada de los métodos habitualmente usados en contextos de producción industrial de software.

La metodología OO-Method ha evolucionado en los últimos trabajos en OO-HMethod [Gómez 2001]. Éste no es más que una ampliación de OO-Method en la que se añade un nuevo modelo para representar la interoperabilidad con los usuarios en sistemas web.

Los principios básicos de OO-Method se concretan en que permite dar soporte a las nociones de modelización conceptual orientado a objetos y usar conceptos de lenguajes de especificación formales y orientados a objetos.

Además, OO-Method posee un avanzado entorno de prototipación automática que incluye tanto la generación automática de una especificación formal y orientada a objetos en OASIS, como la prototipación funcional equivalente al modelo conceptual y la generación completa de código en entornos imperativos como Visual C++ o Delphi.

2. Fases de OO-Method

OO-Method ofrece un marco riguroso para la especificación de sistemas de información que incluye:

- La notación gráfica necesaria para construir el Modelo Conceptual.
- OASIS como un lenguaje de especificación formal.
- La definición de un preciso modelo de ejecución que guía la fase de implementación en el espacio de la solución.

El Modelo Conceptual se compone a su vez en tres modelos:

- El Modelo de Objetos, que utiliza lo que se denomina Diagrama de Configuración de Clases para definir y mostrar la estructura y comportamiento (atributos y operaciones) de todas las clases identificadas en el dominio del sistema. Sería similar al modelo conceptual de EORM u OOHDM (figura 7).
- El Modelo Dinámico, en el que se hace uso de los Diagramas de Transición de Estados y de los Diagramas de Interacción de Objetos, para representar la vida de los objetos y la interacción que se produce entre ellos.
- El Modelo Funcional, en el que, a través de una nomenclatura propia, se recoge la semántica asociada a los cambios de estado de forma fácil e intuitiva.

Si en lugar de trabajar con OO-Method nos movemos en el ámbito de OO-HMethod, a este conjunto de modelos habrá que añadirle el Modelo de Ejecución de Interfaz que servirá para indicar cómo la intefraz y los módulos de la aplicación se comunican entre ellos. Esto se representa a través de lo que se denomina Diagrama de Acceso de Navegación.

Una vez especificado el sistema, se desarrolla un Modelo de Ejecución que establece una representación del modelo conceptual en un entorno de desarrollo, atendiendo tanto a los aspectos dinámicos como estáticos, así como una estrategia de ejecución. Existe en OO-Method un proceso definido para realizar estas estrategias.

Aplicar OO-HMethod u OO-Method a nuestro ejemplo, no resultaría de interés sin hacer uso de su herramienta y esto conllevaría un trabajo descriptivo que se sale del ámbito de este documento. La mayor aportación de estas técnicas es el soporte que el desarrollador encuentra en la herramienta diseñada para trabajar con él.

3. Conclusiones

OO-Method es una técnica bastante adecuada y que cada día está teniendo mayor influencia en el campo del desarrollo de aplicaciones web. Ofrece una importante herramienta web cuyos esfuerzos de elaboración son muy loables.

Sin embargo, OO-Method no aborda tareas como la especificación de requisitos y vuelve a ser una metodología orientada casi en su totalidad a la fase de diseño e implementación.

5.3.8. SOHDM- Scenario-based Objecto-oriented Hypermedia Design Methodology

Otra propuesta algo más reciente que las anteriores es la realizada por H. Lee, C. Lee y C. Yoo: SOHDM [Lee 1998]. Esta propuesta se compone de seis fases y se parece bastante a sus antecesoras RMM, OOHDM y EORM. Sin embargo, hay algo que hace diferente a esta metodología de las anteriores y es el hecho de que se basa en los escenarios para el desarrollo del sistema.

1. Conceptos básicos de SOHDM

Como ya se ha comentado, SOHDM es una metodología para el desarrollo de aplicaciones multimedia que se divide en seis fases que hay que realizar de forma secuencial. Sin embargo, el proceso de desarrollo es un proceso cíclico en el sentido de que al realizar una fase se puede regresar a alguna de las anteriores para refinarla y adaptarla mejor.

Como su propio nombre indica, SOHDM está basado en los escenarios para elaborar las aplicaciones multimedia. En su proceso, los escenarios se elaboran en la fase de análisis para capturar los requisitos funcionales del sistema y sirven como base para el resto del proceso. A pesar de que SOHDM se asemeja bastante a OOHDM y EORM, difiere de ellas en el sentido de que mientras que estas dos metodologías sólo trabajan en la fase de diseño, SOHDM engloba también la fase de análisis.

2. Fases de SOHDM

Ya se ha comentado que SOHDM es un proceso compuesto por seis fases secuenciales. En la figura 19 se presenta un esquema en el que se detallan las fases y los productos obtenidos en cada una de ellas y cómo los productos obtenidos en una fase sirven como referencia para la fase siguiente.

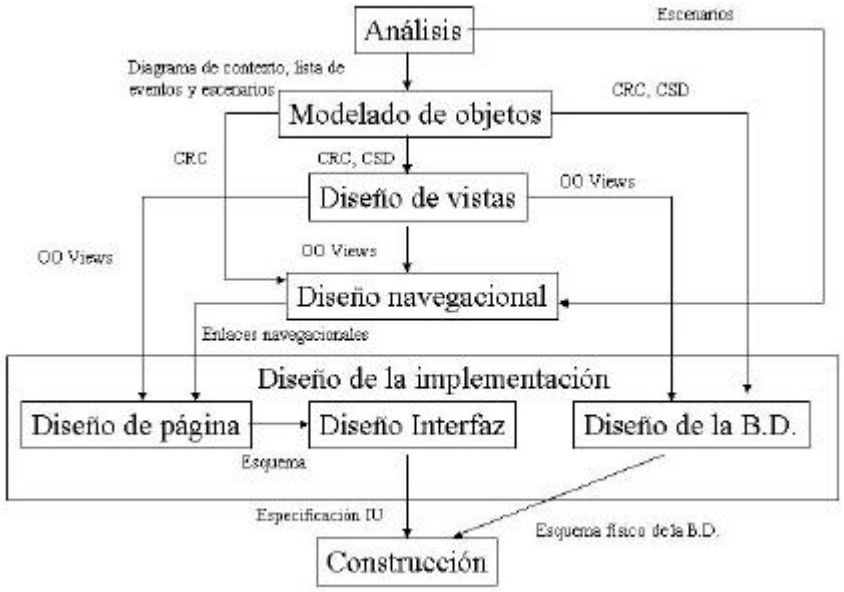

FIGURA 19: ARQUITECTURA DE SOHDM

Sin embargo, a pesar de que el modelo esté pensado como secuencial, permite volver atrás en cualquiera de las etapas del proceso. A continuación se presentarán cada una de estas fases.

Fase 1- Análisis

En la fase de análisis se debe realizar un estudio de las necesidades de la aplicación, del entorno de trabajo y de los actores. La finalidad principal de esta fase es conseguir los escenarios que representen las actividades que se pueden llevar a cabo en el sistema.

Para ello, lo primero que se debe realizar es un diagrama de contexto tal y como se propone en diagramas de flujos de datos [Yourdon 1989]. En este diagrama de contexto es necesario detectar a las entidades externas que se comunican con el sistema, así como los eventos que provocan esa comunicación.

Una vez detectados estos eventos, se debe elaborar lo que se denomina *lista de eventos*. La lista de eventos será una tabla con una estructura similar a la mostrada en la tabla 6.

Nombre de la entidad externa	Nombre del evento
Entidad externa 1	Evento 1 en el que participa la entidad
	...
	Evento n en el que participa la entidad
...	...
Entidad externa m	Evento 1 en el que participa la entidad
	...
	Evento p en el que participa la entidad

TABLA 6: ESTRUCTURA DE LA LISTA DE EVENTOS

En esta tabla se detallan todas las entidades en la columna de la izquierda y en la columna de la derecha todos los eventos en los que cada una participa. Para nuestro ejemplo, la única entidad externa que tenemos es el usuario. Podríamos detectar, por ejemplo que se ve afectado por el evento de conectarse al sistema y el de petición de datos, quedando nuestra tabla como la que se muestra en la tabla 7.

Nombre de la entidad externa	Nombre del evento
Usuario	Conexión al sistema
	Petición de datos

TABLA 7: EJEMPLO DE LISTA DE EVENTOS

Partiendo de esta tabla, por cada evento diferente se debe elaborar un escenario. Éstos se van a representar mediante los denominados SACs[2] (Scenario Activity Chart). En estos escenarios se va a describir el proceso de trabajo que se va a seguir en el sistema cuando se produzca la situación que el escenario representa. Por ejemplo, en la figura 20 nos encontramos un escenario en el que se describe cómo se produce la conexión al sistema del usuario.

2 Los SACs son una forma como cualquier otra de representar los escenarios en la documentación. Esta representación está fuera de los límites de este trabajo, pero se pueden encontrar descritos en [Lee 1998].

Escenario para la conexión al sistema

1. El usuario se conecta al sistema
2. El sistema solicita la clave
3. El usuario introduce la clave
4. El sistema comprueba que la clave es correcta
5. El sistema muestra el menú principal

FIGURA 20: EJEMPLO DE ESCENARIO

Fase 2- Modelado de objetos

Los escenarios representados mediante SACs son usados para conseguir modelar los objetos del sistema. En la fase de modelado de objetos, los escenarios van a ser transformados en objetos según la propuesta de los CRC Cards (Class Responsability Collaboration) [Wirfs-Brock 1990]. Esta propuesta tiene como objetivo presentar un formato sencillo e informal para conseguir un diccionario de datos para las clases del sistema. En ellas cada clase tiene asociado una ficha (CRC Cards) en la que se almacena: su nombre, sus atributos, su superclase, sus subclases, sus componentes, las asociaciones en las que participa, las otras clases con las que colabora y los eventos detallados en los SACs de los que es responsable.

Las CRC Cards irán acompañadas de un diagrama de clases, CSD (Class Structure Diagram), que representará gráficamente lo recogido en las CRC Cards. En principio la nomenclatura seguida fue la de OMT pero ya ha asumido la de UML.

Nuevamente, para nuestro ejemplo, este modelo coincidiría con el de la figura 7.

Fase 3- Diseño de vistas

En la fase de diseño de vistas, los objetos serán reorganizados en unidades navegacionales. Una unidad navegacional representa una vista de los objetos del sistema. Las vistas van a estar muy relacionadas con el concepto de nodos de OOHDM. La vista es una agrupación de información que se presenta agrupada al usuario bajo un determinado criterio.

En SOHDM, las vistas pueden ser:

- Vistas base; son aquellas que toman todos los datos que muestra de una única clase.
- Vistas de asociación; toma los datos de dos clases que se encuentran relacionadas mediante una asociación en el modelo de clases.
- Vistas de colaboración; toma los datos de clases que se encuentran relacionadas mediante una relación de colaboración.

Fase 4- Diseño Navegacional

En el diseño navegacional se van a definir los enlaces o hiperenlaces que existen entre las diferentes vistas. Aquí las vistas definidas en la fase anterior se relacionan a través de estructuras de acceso.

El primer paso a realizar es definir lo que se conoce como *nodos de estructuras de acceso* (ASNs). Estos son nodos especiales como diccionarios, menús, etc. que sirven como punto de partida para la navegación. Una vez definidos los ASN, estos y las vistas definidas en la fase anterior, se conectan mediante flechas que indican el sentido de la

navegación. Se obtiene así un grafo en el que se representa como se navega desde un punto de información (vista o ASN) a otro. A este modelo se le denomina *enlaces navegacionales*.

A pesar de que el grafo que surge es bastante intuitivo, SOHDM propone una alternativa para representar los *enlaces navegacionales.* Por cada componente conexa de este grafo, se elabora una matriz, denominada *matriz de enlaces navegacionales*. En esta matriz se indica como los nodos de esa componente se relacionan entre sí[3].

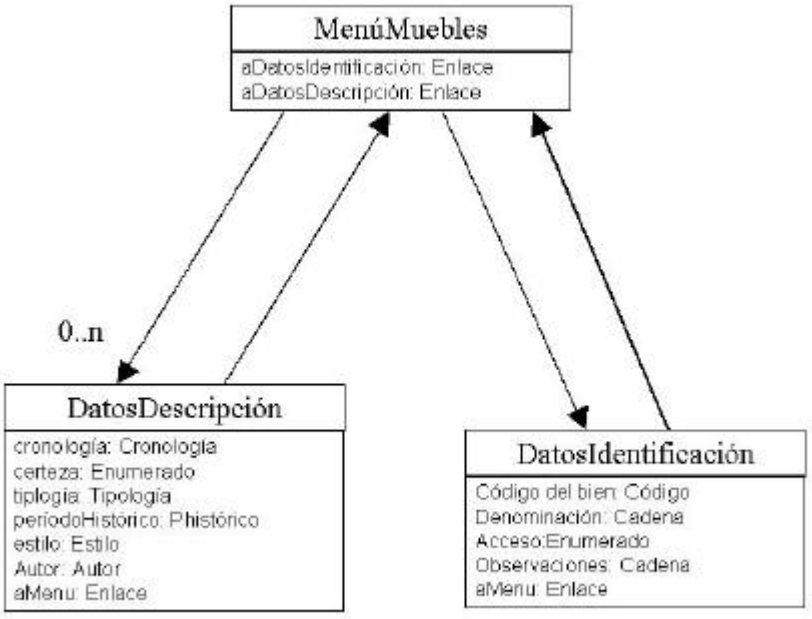

FIGURA 21: EJEMPLO DE MODELO DE DISEÑO NAVEGACIONAL DE SOHDM

El grafo resultante de esta fase y de la anterior para nuestro ejemplo, puede recordarnos bastante al modelo de clases navegacionales de OOHDM. El concepto es el mismo, cambia en las denominaciones: el nodo pasa a llamarse vista y las clases índices a estructuras de acceso y los enlaces son direccionales. En la figura 21 vemos el resultado para nuestro ejemplo. Vemos que desde el menú podemos navegar a los datos de identificación del bien y a los datos de descripción. Desde estos se puede volver al menú. Nótese que al tener un bien varios valores para los datos de descripción, la multiplicidad en este enlace es de 0..n.

Fase 5- Diseño de la implementación

En esta fase se van a generar esquemas de páginas que van a representar los puntos de información definidos en la fase anterior dentro de un entorno determinado. Para cada esquema se debe indicar: su nombre, su título, las vistas que engloba, una breve descripción de su significado y una lista con los enlaces que tiene.

[3] No se verá la nomenclatura de representación de la matriz de enlaces navegacionales, por ser un poco sofisticada. Se encuentra detallada en [Lee 1998]

Tras definir estos puntos de información se hace un diseño de la interfaz de usuario. SOHDM tiene prevista una nomenclatura normalizada para representar los posibles elementos que se pueden encontrar en una pantalla: botones, imágenes, listas, etc.

Una vez definida la interfaz de usuario y los esquemas, es necesario definir la base de datos. En principio las aplicaciones hipermedia deben definirse en sistemas gestores orientados a objeto, pero, acercándose más a la realidad, permite el uso de sistemas gestores de bases de datos relacionales. Para poder llevar una representación orientada a objetos a un modelo relacional, es necesario aplicar técnicas de conversión. Los autores de SOHDM proponen las detalladas en [Fong 1995] y [Blaha 1988].

Fase 6- Construcción

En la fase de construcción se debe implementar una aplicación hipermedia ejecutable en función de las pantallas y las páginas definidas en la fase anterior. Igualmente debe desarrollarse la base de datos física para soportar la aplicación.

3. Conclusiones

SOHDM es hasta ahora la única propuesta que tiene en cuenta aspectos como la especificación de requisitos haciendo uso de los escenarios. Es una propuesta bastante interesante pues cubre todas las fases del proceso de desarrollo, obviando la implantación y las pruebas.

SOHDM es una propuesta joven que no ha sido muy usada aún. Tiene como ventaja que es un proceso sencillo de seguir, aunque se le puede criticar el hecho de que su nomenclatura está muy cerrada. Por ejemplo, para el desarrollo de la interfaz se define cómo se representa una imagen o un botón en el modelo, aunque no se dice nada de cómo se representa un elemento de audio, sin dejar ninguna opción a que el diseñador pudiese definir su propia representación.

5.3.9. RNA: Relationship-Navigational Analisys

1. Conceptos básicos de RNA

RNA fue propuesto por Bieber, Galnares y Lu en 1998 [Bieber 1998]. RNA plantea una secuencia de pasos para el desarrollo de aplicaciones web, centrándose fundamentalmente en la fase de análisis.

Aunque la propuesta está orientada al desarrollo de aplicaciones que traten temas jurídicos o de leyes, sus ideas pueden aplicarse en otros entornos.

Como veremos a continuación, RNA no hace nuevas propuestas de modelos o técnicas para desarrollar el proceso. Lo que RNA ofrece es una secuencia de pasos a seguir, sin indicar cómo hacer cada uno de ellos.

2. Fases de RNA

Fase 1- Análisis del entorno

El propósito de esta fase es el de estudiar las características de la audiencia. Consiste en determinar y clasificar en grupos al usuario final de la aplicación según sus perfiles.

Fase 2- Elementos de interés

En esta fase se listan todos los elementos de interés de la aplicación. Por elementos de interés se entienden los documentos, las pantallas que se van a requerir, la información, etc. RNA da el objetivo de esta fase de esta forma tan genérica. No especifica cómo se deben listar estos elementos, sólo indica que hay que listarlos.

Fase 3- Análisis del conocimiento

Aquí hay que desarrollar y plantear un esquema que represente a la aplicación, hay que identificar los objetos, los procesos y las operaciones que se van a poder realizar en la aplicación y las relaciones entre estos.

Fase 4- Análisis de la navegación

En esta fase se enriquece el esquema obtenido, basándose en las relaciones y objetos definidos en la fase anterior, con las posibilidades de navegación dentro de la aplicación.

Fase 5- Implementación del análisis

Una vez obtenido el esquema final en el que ya se encuentran incluidos los aspectos de navegación, se pasa el esquema a un lenguaje entendible por la máquina.

3. Conclusiones

Como ya se enunciaba, RNA no ofrece ningún añadido en cuanto a nuevos modelos o técnicas. Sólo da una guía de desarrollo, indica qué pasos hay que realizar sin indicar cómo realizarlos.

Sin embargo, el hecho de que se haya presentado aquí se debe a que sí que puede ser interesante a la hora de plantear metodologías para el desarrollo de aplicaciones, sobretodo en lo que sería la fase de análisis. RNA marca la importancia que en las aplicaciones web tiene el estudio de los usuarios (fase 1), de los conceptos clásicos (fase 3) y de la navegación (fase 4). Plantear estos tres aspectos de forma autónoma, estudiándolos por separado en diferentes fases, resulta una propuesta muy extendida y muy conveniente para el desarrollo de estas aplicaciones. Además es la única propuesta que marca como importante el estudio del entorno y de los elementos de interés para conocer el ámbito y alcance del problema antes de abordarlo.

5.3.10. HFPM: Hypermedia Flexible Process Modeling Strategy

1. Conceptos básicos de HFPM

HFPM fue propuesta por Luis Olsina en 1998 [Olsina 1998]. Es la única de las metodologías propuesta en este apartado que engloba todas las fases del proceso de desarrollo, va desde el análisis hasta el desarrollo de la documentación y el mantenimiento. Además divide y detalla cada una de las tareas que comprende cada fase.

Sin embargo, a pesar de que se den las tareas y subtareas a realizar, esta propuesta no ofrece nuevos modelos o técnicas de modelado, por lo que no se aplicará al ejemplo con el que estamos trabajando. Aún así es bastante interesante de estudiar por la estructuración del proceso de desarrollo que presenta.

A continuación veremos cada una de las fases de forma general.

2. Fases de HFPM

Fase 1- Modelado de los requisitos del software

En esta fase se proponen las siguientes tareas:

- Descripción breve del problema.
- Descripción de los requisitos funcionales mediante los casos de uso.
- Realizar un modelo de datos para esos casos de uso.
- Modelar la interfaz de usuario.
- Modelar los requisitos no funcionales. En éstos incluyen la navegación, la seguridad, etc.

En principio no da ninguna norma a seguir para realizar estas tareas, dejando flexible las representaciones.

Fase 2- Planificación

En esta fase se plantea el plan de trabajo del proyecto, se analiza y se especifica de forma concisa. No se presenta ningún patrón o proceso a seguir para ello.

Fase 3- Modelado conceptual

El objetivo de esta fase es similar al objetivo de la fase de modelado conceptual de OOHDM. En esta fase se debe conseguir un modelo de clases que represente al sistema sin entrar en aspectos de hipermedia. Las tareas a realizar son:

- Analizar el dominio del problema.
- Modelar el sistema mediante un diagrama de clases.

Fase 4- Modelado Navegacional

Está íntimamente basado en las propuestas de OOHDM y EORM. En esta fase se trata de conseguir un modelo navegacional que represente las posibilidades de navegación del sistema. Para ello, es necesario:

- Analizar las necesidades de los usuarios.
- Identificar las clases navegacionales, entendiendo por clases navegacionales las propuestas por OOHDM.
- Especificar el esquema de navegación y las clases de navegación.
- Analizar los contextos de navegación.
- Especificar los contextos navegacionales.

Fase 5- Modelado de la Interfaz Abstracta

Consiste en hacer un diseño de la interfaz sin entrar en características propias del lenguaje de programación. Las tareas que se proponen son:

- Analizar las necesidades de interfaz de usuario.
- Detectar los objetos, los eventos y los enlaces visibles para cada usuario.
- Diseñar los prototipos.

Fase 6- Diseño del Entorno

Esta fase tiene por objetivos enriquecer los modelos obtenidos mediante el uso de patrones de diseño. Además es en esta fase en la que se decide la arquitectura del sistema y la división en subsistemas. Las tareas son las siguientes:

- Usar patrones de diseño para mejorar los modelos.
- Diseñar la arquitectura, si es posible haciendo uso de patrones.
- Dividir el sistema en subsistemas.

Fase 7- Capturar y editar los elementos multimedia

En esta fase se deben plantear los múltiples medios con los que se va a trabajar, así como los sistemas de almacenamiento que se usarán en los mismos.

Fase 8- Implementación

Esta fase tiene por objetivo conseguir el programa ejecutable que represente la aplicación. Al igual que la fase anterior no tiene tareas.

Fase 9- Verificación y validación

En esta fase se va a analizar si el resultado es adecuado en base a los requisitos estudiados en la primera fase del proceso. No se dan pautas a seguir para ello dentro de la propuesta de HFPM.

Fase 10- Evaluación del entorno

Aquí se analiza si el resultado se adecúa al entorno en el que se va a implantar.

Fase 11- Evaluación de la calidad

Se debe evaluar y determinar si el producto resultante es de alta calidad. En este aspecto HFPM no especifica qué entiende por calidad ni cómo se puede evaluar la calidad del producto.

Fase 12- Mantenimiento

En general esto implica tanto el mantenimiento correctivo, aumentativo y adaptativo del sistema, aunque nuevamente no se dan pautas a seguir para ello.

Fase 13- Documentación

Como resultado final se debe generar la documentación del sistema. Dicha documentación engloba:

- La documentación de cada uno de los modelos generados en cada fase
- La documentación del resultado de las pruebas y las valoraciones realizadas en las fases 9, 10 y 11.
- El manual de usuario.

Aunque la presentación de las fases es secuencial, HFPM permite la vuelta atrás y la realización de iteraciones en el proceso de desarrollo.

3. Conclusiones

HFPM ofrece las pautas que se deben seguir para el desarrollo de las aplicaciones multimedia. Sin embargo, y como se puede deducir del apartado anterior, no ofrece una metodología detallada. Sólo indica qué se debe hacer y no cómo se debe hacer.

También puede observarse que se basa bastante en OOHDM y en las metodologías clásicas de la orientación a objetos (OMT, UML, etc.). En definitiva HFPM integra las propuestas clásicas de la orientación a objetos con la de OOHDM, reutiliza los modelos presentados por éstas y ofrece una directriz clara a seguir en el proceso de desarrollo.

5.3.11. OO/Pattern Approach

1. Conceptos básicos de OO/Pattern Approach

La aproximación que veremos en este apartado fue propuesta por Thomson, Greer y Cooke en 1998 [Thomson 1998]. Esta propuesta es bastante similar a HFPM pues ambas proponen el uso de patrones y de la orientación a objetos para el diseño navegacional y la interfaz. Sin embargo, esta propuesta, a diferencia de HFPM, no cubre el ciclo completo de desarrollo.

Algo característico de esta propuesta es el hecho de que utiliza los conocidos casos de uso para realizar la fase de análisis de la aplicación. Veamos cada una de estas fases.

2. Fases de OO/Pattern Approach

Fase 1- Diseño de los casos de uso

Basándose en la técnica de los casos de uso, tal y como la propone UML [Jacobson 1999], esta metodología propone capturar los requerimientos del sistema que servirán para realizar el posterior diseño de la aplicación. En la figura 20 vemos uno de los posibles casos de uso para nuestro ejemplo. Mediante este caso de uso representaríamos la opción del usuario a dar de alta.

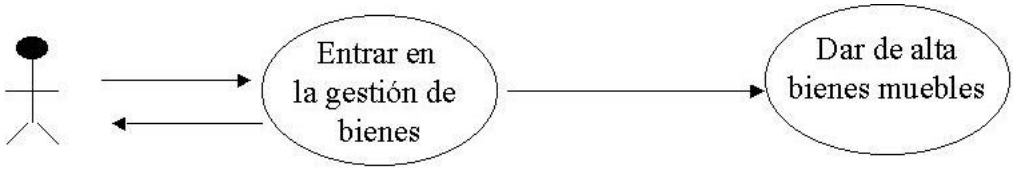

FIGURA 22: EJEMPLO DE CASO DE USO

Fase 2- Diseño conceptual

Partiendo de los casos de uso, se debe modelar la aplicación sin entrar en aspectos de interfaz o de navegación. Se pretende conseguir un modelo de clases que represente al sistema. Para ello, se deben aplicar las técnicas de la orientación a objetos. En este caso, el resultado aplicado a nuestro ejemplo sería equivalente al que se mostró en la figura 7.

Fase 3- Diseño de colaboración

En esta fase es necesario aplicar la técnica de los diagramas de colaboración propuesta por UML [Jacobson 1999], para representar las relaciones entre las clases del sistema.

Fase 4- Realizar el diccionario de datos para modelo de clases

En esta fase se debe documentar los modelos obtenidos en las dos fases anteriores. En principio el formato de este diccionario es libre.

Fase 5- Diseño Navegacional

En esta fase se debe enriquecer el modelo de clases con nuevas clases que permitan representar la navegación. Para ello, se definen una serie de patrones de diseño en los que se definen los nodos y los enlaces de forma similar a como se definieron en EORM o OOHDM. El resultado sería un modelo de clases navegacionales similar al de OOHDM. Así el de nuestro ejemplo sería equivalente al que se obtuvo en OOHDM y que se mostró en la figura 9.

Fase 6- Implementación

La fase de implementación tiene por objetivo el conseguir, basándose en el modelo navegacional, la aplicación ejecutable del sistema.

3. Conclusiones

Esta metodología es bastante completa en el sentido de que hace referencia a la fase de análisis, diseño e implementación. Sin embargo no deja muy claro qué documentación hay que presentar y no hace ningún tipo de referencia a la interfaz. Sin embargo hay que resaltar que establece la necesidad de realizar un diccionario de datos para el modelo conceptual.

Por ello, de HFPM podemos concluir en que es una propuesta que recoge todo el ciclo de vida del sistema mostrando así que esto es necesario. Sin embargo, no trata aspectos básicos como la interfaz ni indica qué hay que hacer en cada fase. Sólo recoge qué objetivos deben alcanzarse en cada una de ellas.

5.3.12. El Proceso Unificado

1. Conceptos básicos del Proceso Unificado

Como ya se ha comentado, UML es una propuesta de lenguaje de modelado de datos realizada por Booch, Rumbaugh y Jacobson, entre otros. La primera versión de UML nace en 1997.

UML es un lenguaje gráfico para modelar sistemas software según la orientación a objetos, en él se describen una serie de modelos que nos permiten representar diferentes aspectos de nuestros sistemas software. No es objetivo de este trabajo el describir las posibilidades de modelado que ofrece UML [Booch 1999] [Rumbaugh 1999].

En base a UML, los mismos autores realizan una propuesta de metodología denominada Proceso Unificado [Jacobson 1999], que es la que se verá en este documento. El Proceso Unificado comprende un conjunto de actividades que hay realizar para llevar a cabo el desarrollo de producto software. A continuación vamos a describir esta propuesta.

2. Fases del Proceso Unificado

El ciclo de vida del Proceso Unificado es el ciclo de vida que los autores creadores de UML proponen para el desarrollo. El ciclo de vida del Proceso Unificado es en la práctica un ciclo de vida en espiral. Sus características esenciales es que se trata de un ciclo de vida incremental e iterativo. Iterativo porque se producen varios ciclos en su desarrollo en cada uno de los cuales se concreta más el producto resultado del ciclo anterior. E incremental porque en cada ciclo, el producto resultante se va a adecuar más a las necesidades de los clientes.

El ciclo de vida del Proceso Unificado asume que en la vida del proyecto existen una serie de ciclos, cada uno de los cuales tiene cuatro fases. Por cada una de las fases se pueden realizar varias iteraciones y en éstas a su vez se distinguen cinco flujos de trabajo. Vemos un esquema de ello en la figura 23. En esta figura se usa la propia notación de UML para representar la estructura del ciclo de vida del Proceso Unificado.

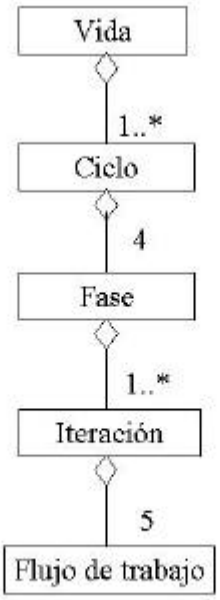

FIGURA 23: CICLO DE VIDA DEL PROCESO UNIFICADO

Tras cada ciclo, obtendremos un resultado final que será una versión del producto.

Como ya hemos dicho, en cada ciclo existen 4 fases que se detallan en la figura 24.

Como vemos, tras cada una de las fases obtenemos un resultado de fase. El resultado de la última fase va a coincidir con una versión del producto. A continuación se presenta un pequeño resumen de lo que hay que realizar en cada una de ellas.

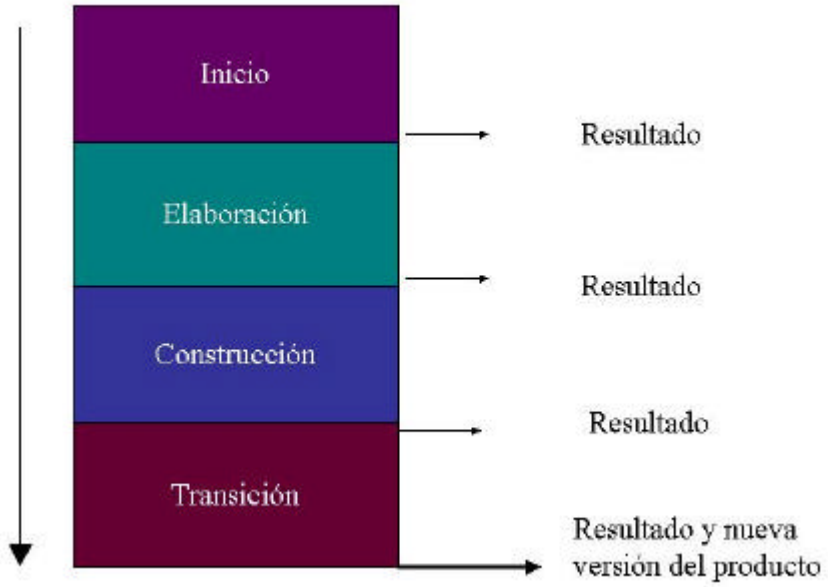

FIGURA 24: FASES DEL PROCESO UNIFICADO

Fase 1-Fase de Inicio

En ella las tareas que fundamentalmente se deben llevar a cabo son:

- Estudiar la viabilidad del ciclo.
- Identificar riesgos.
- Estimar costes y realizar la planificación.
- Perfilar la arquitectura del sistema.

Fase 2-Fase de Elaboración

En esta fase, debemos:

- Revisar la identificación de riesgos de la etapa anterior.
- Especificar los requisitos de usuario.
- Definir la arquitectura del sistema.
- Preparar el plan de Proyecto.
- Revisar la planificación y estimación de costes.
- Definir parámetros generales como calidad, fiabilidad, riegos, tiempos de respuesta, etc.

Fase 3-Fase de Construcción

En la fase de construcción, se debe:

- Completar la arquitectura.
- Obtener el código ejecutable.
- Obtener el producto beta.

Fase 4-Fase de Transición

La fase de transición es la última fase del ciclo y tras ella obtendremos una versión del producto. Sus tareas fundamentales son:

- Realizar pruebas beta y subsanar deficiencias.
- Formación del personal, mantenimiento y garantía.

Por su parte, hemos dicho que en cada fase hay N iteraciones. Tras cada iteración se obtiene los que se conoce como producto interno. El producto interno es algo que, a diferencia de los resultados de fase, no suele darse al usuario. El esquema de las iteraciones lo vemos en la figura 25.

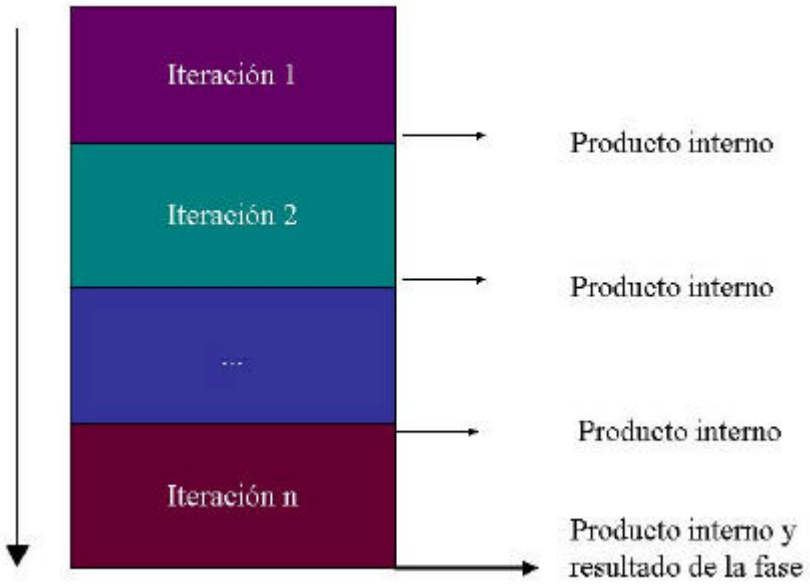

FIGURA 25: ITERACIONES DEL PROCESO UNIFICADO

Por su parte, en cada iteración hay cinco flujos de trabajo: requisitos, análisis, diseño, implementación y pruebas. Realmente es como si dentro de cada iteración elaborásemos un ciclo de vida secuencial, puesto que estos flujos de trabajo son similares a los propuestos por el ciclo de vida clásico.

Los flujos de trabajo van a ir tomando más o menos importancia dependiendo de la fase en la que estemos trabajando.

Veamos a continuación, muy brevemente, cuáles son los objetivos de cada uno de estos flujos de trabajo:

?? Captura de requisitos: el propósito general de este flujo de trabajo es dirigir el proceso de desarrollo hacia el sistema correcto. Para ello, UML propone el uso de los casos de uso [Jacobson 1995]. Básicamente destaca la necesidad de capturar los requisitos de almacenamiento, los requisitos funcionales y los requisitos no funcionales.

?? Análisis: se analizarán los requisitos descritos en el flujo anterior para refinarlos y estructurarlos. El propósito de hacer esto es alcanzar una comprensión mayor del problema permitiendo así que se pueda dar una visión de la estructura completa del

sistema. Esto se representa principalmente a través del diagrama de clases y del diagrama de paquetes [Booch 1999].

?? Diseño: se debe dar forma al sistema en función del modelo de análisis y de los requisitos establecidos. En este flujo se determinará la arquitectura y la división en subsistemas, pero básicamente la idea es conseguir una representación abstracta del sistema que se acerque mucho a la implementación pero sin entrar en detalles de bajo nivel.

?? Implementación: partiendo del resultado de la fase de diseño, en la implementación hay que llevar a código entendible por la máquina, todas las características representadas y capturadas en los flujos anteriores. En definitiva se deben implementar los subsistemas y las clases, los interfaces y las relaciones, de manera que consigamos una aplicación que represente al sistema y cumpla los requisitos establecidos en el primer flujo de trabajo.

?? Pruebas: el objetivo fundamental es verificar el resultado de la implementación. Para ello, se debe diseñar un test de pruebas que examine la corrección de cada una de las unidades de programación.

3. Conclusiones

Si se desea desarrollar un sistema de información global y se decide usar el Proceso Unificado descrito en el apartado anterior, se detectarán las carencias que presenta esta propuesta para representar aspectos como la interfaz de usuario, la multiplicidad de medios en los tipos de información o la complejidad de la navegación dentro del sistema. Esto se debe a que UML y el Proceso Unificado están orientados a trabajar esencialmente con la funcionalidad y los aspectos de almacenamiento de informacón

Detectada esta carencia, diferentes grupos de investigación han intentado resolver el problema enriqueciendo a UML con nuevos recursos para poder representar estos aspectos propios de las aplicaciones en la web. Dentro de estos grupos, se pueden determinar dos tendencias.

La primera de ellas englobaría a los investigadores que intentan enriquecer UML añadiendo nuevos esterotipos que permitan representar aspectos propios de las aplicaciones web. En este sentido vamos a estudiar la propuesta realizada por Conallen [Conallen 1999a] en el apartado 5.3.13.

Por otro lado, existen otra serie de propuestas que, basadas en metodologías de desarrollo de sistemas multimedia, en especial OOHDM, intentan enriquecer a UML no a través de estereotipos, sino a través de nuevos modelos que permitan representar la multimedia y la navegación. Estas propuestas nacen fundamentalmente en Alemania y las analizaremos en el apartado 5.3.14.

A pesar de que al intentar aplicar el Proceso Unificado al diseño de los sistemas de información global encontremos lagunas a la hora de expresar algunos aspectos, está claro que la nomenclatura de UML y su ciclo de vida es un estándar que está dando muy buenos resultados. De hecho, la mayoría de las propuestas que hemos analizado en este trabajo hacen uso de él.

5.3.13. Building Web Applications with UML

1. Conceptos básicos de "Building Web Applications with UML"

Como ya se ha comentado, de manos de Jim Conallen aparece una propuesta [Conallen 1999a] que no es más que una ampliación de UML para la web. Los creadores de UML reconocieron que UML no era una propuesta perfecta para todo los tipos de aplicaciones, así que junto a él definieron un camino formal para la definición de nuevos conceptos que permitieran cubrir necesidades en sistemas novedosos. Conallen ha hecho uso de esta propuesta y, observando las carencias que UML presentaba para diseñar aplicaciones en la web, realizó una propuesta para ampliar el lenguaje gráfico con nuevos estereotipos propios de internet y la web.

Los estereotipos propuestos por Conallen usan unas representaciones gráfica que él mismo pone a disposición del interesado y que se pueden incluir en la herramienta Rational Rose.

A continuación se detallarán estos estereotipos, aunque no se detallarán de forma exhaustiva. Si se desear ver la definición formal de éstos puede consultarse en la web: http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm [Conallen 1999b].

?? Clases

Las clases que Conallen propone en su extensión son:

a) Server Page: representa una página web que contiene uno o varios scripts que van a ser ejecutados por el servidor. Las operaciones de los objetos de esta clase representan las funciones de los scripts y sus atributos las variables que son visibles en el ámbito de la página. Su representación la vemos en el icono 1 de la figura 26.

b) Client Page: una instancia de este tipo de clases es una página web escrita en HTML. Como cualquier página web es una mezcla de datos, presentación y lógica. Son páginas representadas y leídas por los navegadores. Puede poseer scripts, pero estos serán interpretados por los navegadores. En el icono 2 de la figura 26 vemos su representación.

c) Form: una clase Form se traduce directamente a una etiqueta form en HTML. Representa a un conjunto de campos de entrada que podemos encontrarnos dentro de una página. Sus atributos se corresponderán con los atributos de los forms representados en la página web. No posee operaciones. Cualquier operación que interactúe con el Form será una propiedad de la página que lo contenga. Su representación la vemos en el icono 3 de la figura 26.

d) Frame Set: un Frame Set es un contenedor de múltiples páginas web. Es decir un Frame Set va a contener páginas web u otros Frame Set. Éstos se traducen directamente a frameset de HTML. Vemos su representación en el icono 4 de la figura 26.

e) Target: representa un compartimento en una ventana de un navegador donde puede ser interpretada una página web. Básicamente es un frame en una ventana definida por un frameset, aunque también podría ser una nueva instancia del propio navegador. Vemos en el icono 5 de la figura 26 cómo se representa.

f) JavaScript: esta clase sirve para representar los objetos definidos por el usuario dentro de una página web mediante una aplicación JavaScript. Vemos su representación en el icono 6 de la figura 26.

?? Asociaciones

a) Link: representa un enlace desde una página a otra. Se traduce directamente a una etiqueta de HTML del tipo anchor.

b) Targeted Link: es similar al anterior pero aquí la página destino y la página origen es la misma. Nos movemos entre etiquetas dentro de la misma página.

c) Frame Content: es una relación de agregación donde el contenedor es un objeto de la clase FrameSet y los contenidos son otras páginas.

d) Submit: se da siempre entre un objeto de la clase Form y uno de la clase Server Page. Sirve para representar que ese Form interactúa con ese servidor.

e) Builds: este tipo de asociaciones se da entre objetos de la clase Server Page y objetos de la clase Client Page. Sirve para identificar qué página servidor es responsable de la creación de una determinada página cliente.

f) Redirect: es una asociación unidireccional. Sirve para representar que la página destino es cargada inmediatamente después de la página origen.

g) Object: se establece entre una página cliente y un objeto que se encuentra en ella.

h) IIOP (Internet Inter-Orb Protocol): es una relación especial que se produce entre objetos en el servidor y objetos en el cliente. Representa un mecanismo de comunicación (diferente a HTTP) para las comunicaciones entre el cliente y el servidor.

i) RMI (Remote Method Invocation): RMI es un mecanismo de los Applets y los Beans de Java para mandar mensajes a otros Beans en otras máquinas. Básicamente esta relación se establece entre objetos Java Beans o Applets en el cliente y Java Beans en el servidor.

?? Atributos

a) Input Element: es un atributo de la clase Form que se corresponde con campos de entrada en el Form de la página HTML.

b) Select Element: es un atributo de la clase Form que representa un campo cuyo valor, o valores, pueden ser seleccionados a partir de los elementos de una lista. Básicamente se traduce a los controles ComboBox y ListBox.

c) Text Area Component: es un atributo de la clase Form que representa a un área de texto.

?? Componentes

a) Page: un componente Page representa a cualquier tipo de página web. El icono mediante el que se representa lo vemos en el icono 7 de la figura 26.

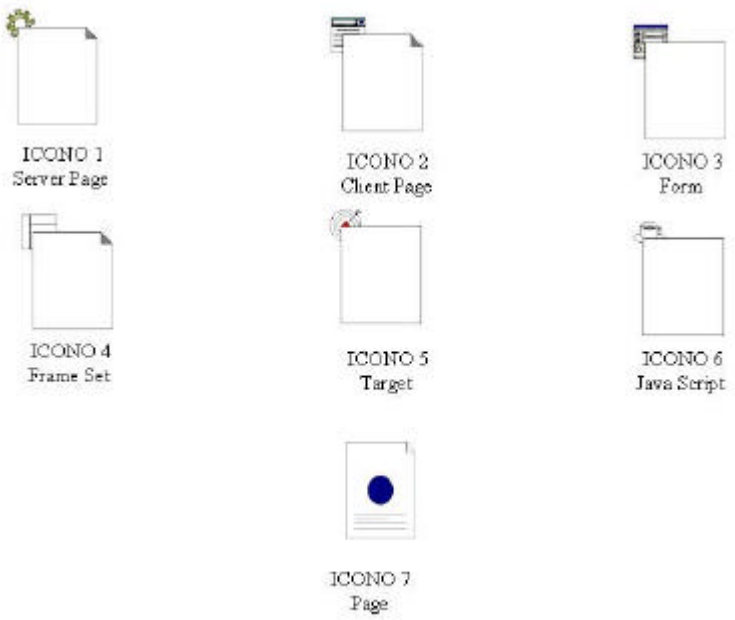

FIGURA 26: ICONOS PROPOPUESTOS POR CONALLEN

2. Conclusiones

Como ya se enunció, esta propuesta simplemente define estos nuevos estereotipos que se pueden incluir en las representaciones de los sistemas web. Sin embargo, y a pesar de que esta extensión de UML nos ofrece semántica para representar conceptos como las páginas web o los enlaces, no nos ofrece ninguna guía de cómo representar la información almacenada en múltiples medios, la navegación o la interfaz de usuario. Estos estereotipos son muy cercanos a la fase de implementación y recogen aspectos como el hecho de que un script este escrito en Java o no. Los sistemas de información global no son, sin embargo, diferentes sólo en el bajo diseño y en la implementación a los sistemas clásicos. Sus diferencias deben ser tratadas desde las primeras fases y esta extensión no nos ofrece herramientas suficientes para ello.

5.3.14. Specification and modeling of multimedia and hypermedia systems

1. Conceptos básicos de *"Specification and modeling of multimedia and hypermedia systems"*

La segunda propuesta para la ampliación de UML que vamos a analizar difiere de la anterior en el sentido de que ofrece la descripción de nuevos modelos para representar los aspectos de las aplicaciones multimedia y de la web. Esta propuesta se basa en tres pilares:

- OOHDM [Schwabe 1995] (Object Oriented Hypermedia Design Model), que vimos en el apartado 5.3.5.
- YAON [Maier 1997] (Yet Another Object Notation), es una notación para desarrollar documentos gráficos que permite recoger las decisiones de implementación bajo el paradigma de la orientación a objetos. Está orientado principalmente a los sistemas distribuidos.
- EPK-fix [Knapp 1997], es una metodología de desarrollo para catálogos multimedia. En ella un catálogo se ve como una aplicación multimedia que se

desarrolla en base a cuatro componentes: la *estructura*, el *layout*, la *dirección* y los *servicios*. Cada componente interactivo es una representación de un producto de la base de datos de los productos del catálogo. Mediante la estructura se representa la forma de la base de datos, el esqueleto de la aplicación. Con el layout se describen los aspectos estáticos del catálogo (ventanas, frames, etc.) . Todos los aspectos de navegación, ya sea entre los elementos del propio catálogo como entre los elementos del layout, son representados mediante la dirección. Y por último a través de los servicios se representarán otras utilidades que ofrezca el sistema: menús de ayuda, enlaces de interés, etc. A pesar de que EPK-fix está totalmente orientado a la generación de catálogos multimedia, muchas de sus ideas han sido extendidas en la propuesta que se está analizando.

2. Fases de *"Specification and modeling of multimedia and hypermedia systems"*

La extensión de UML que se va a presentar en este apartado, viene de manos de Nora Koch de la Universidad de Munich y Luis Mandel de la Industria de Tecnología Software de Alemania [Mandel 2000]. En esta propuesta, se detallan tres subetapas de diseño dentro del flujo de diseño: diseño conceptual, diseño navegacional y diseño de la presentación. En cada uno de ellos se proponen nuevos modelos y estereotipos que se verán a continuación.

?? Diseño Conceptual

El principal objetivo del diseño conceptual es capturar el dominio semántico del problema sin entrar en aspectos de navegación o de interfaz. Las tareas básicas a realizar son:

1. Determinar las clases
2. Especificar los atributos y operaciones de las clases
3. Definir las estructuras jerárquicas
4. Determinar los subsistemas

Básicamente se correspondería con la propuesta inicial de diseño del Proceso Unificado. El resultado es un diagrama de clases clásico que se denomina *modelo conceptual*. Así para nuestro ejemplo sería similar al que se vio en la figura 7.

?? Diseño Navegacional

El diseño de la navegación supone un paso crítico en el diseño de cualquier aplicación hipermedia, incluso en las más simples. Con el diseño navegacional se va a representar la estructura de enlaces de la aplicación web. Para ello se van a construir dos modelos complementarios: el modelo de clases navegacionales y los diagramas de contextos. Estos modelos van a definir lo que se conocen como *clases navegacionales*. Las clases navegacionales pueden ser de tres tipos: nodos, que representan contenedores de clases; enlaces, que representan la posibilidad de navegar desde un nodo a otro; y estructuras de acceso, que sirven para representar menús, diccionarios, etc. A continuación se verán estas clases a medida que se describen los modelos.

- **Modelo de clases navegacionales**: El modelo de clases navegacionales va a definir una vista del modelo conceptual definiendo qué clases del modelo conceptual pueden ser visitadas en la navegación. Este modelo se construye con un conjunto de clases de navegación y asociaciones entre ellas. Cada

clase y cada asociación del modelo navegacional se define en base a una clase o asociación del modelo conceptual. Se podría decir que el modelo de clases navegacionales es un subgrafo del modelo conceptual en el que se han eliminado algunas clases y atributos indiferentes para la navegación.

El primer paso para construir este modelo es determinar qué clases del modelo conceptual van a aparecer en la navegación y cuáles de sus atributos van a ser visibles. Con esto se obtendrá un nuevo diagrama de clases en el que cada clase del modelo de clases navegacionales será un nodo que habrá que definir en base a las clases del modelo conceptual siguiendo la sintaxis mostrada en la figura 27.

Donde:

- nombreNodo: es el nombre de la clase nodo que se está creando.
- nombreClase: es el nombre de la clase de la clase del modelo conceptual de la que el nodo toma información.
- atributo-i: representa al atributo i-ésimo del nodo.
- nombre-i y nombreVar-i: son alias que se asignan a las clases conceptuales o a los atributos de éstas para representarlos en la expresión de selección.
- expresiónLógica: es un predicado que depende de las variables y clases conceptuales.
- timeout: es un entero que especifica la cantidad de tiempo que se esperará para cargar la página.
- refresh: es un enlace a otro nodo que se carga cuando ha pasado el timeout.
- expires: indica la fecha en la que caduca la información contenida en el nodo.

```
NODE nombre [FROM nombreClase: nombreVar][INHERITS FROM
claseNodo]

atributo1:tipo1 = [SELECT nombre1]
                  [FROM clase1: nombreVar1,... ,clase-n: nombreVar-n]
                  [WHERE expresiónLógica]
atributo2:tipo2 = [SELECT nombre2]...
...

atributo-n:tipo-n =[SELECT nombre-n]...

refresh: Link
timeout: Integer
expires: Date
END
```

FIGURA 27: SINTAXIS DE DEFINICIÓN DE NODOS

Esta idea puede recordar a la que vimos al enunciar OOHDM. La verdad es que es bastante similar. El modelo navegacional para nuestro ejemplo sería el mismo que vimos en el apartado 5.3.5. en la figura 9. Con respecto a la definición del nodo sería bastante similar. Por ejemplo, para la clase DatosIdentificación quedaría como se muestra en la figura 28.

Aquí por ejemplo, vemos que Datos de identificación tendrá cuatro atributos que obtiene de la clase BienMueble del modelo conceptual. Indica que cuando el timeout, que como ejemplo se ha asignado a 5 minutos, expira, se pasa a la clase BienMueble. Nótese que esta clase BienMueble es del modelo navegacional y no del conceptual. Por último, como ejemplo se indica que la información se supone válida hasta el 31 de diciembre de 2020.

```
NODE DatosIdentificacion [FROM BienMueble:BM]

        Código: entero = Select código from BM

        Denominación: cadena = Select denominación from BM

        Acceso: cadena = Select acceso from BM

        Observaciones: cadena = Select observaciones from BM


        refresh: BienMueble
        timeout: 5 minutos
        expires: 31/12/2020
END
```

FIGURA 28: EJEMPLO DE DEFINICIÓN DE NODO

Con el modelo de clases navegacionales y la definición formal de cada nodo estaría realizada solo una parte del diseño navegacional. El otro modelo, es el diagrama de contexto que veremos a continuación.

- **Diagrama de contexto**: El diagrama de contexto está formado por un conjunto de contextos navegacionales. Mientras que el diagrama de clases navegacionales indica cómo se muestra la información en la navegación, en el contexto navegacional se muestra cómo se accede a ella. Normalmente un contexto navegacional comienza con una pregunta del usuario y representa a través de un grafo como se navega en el sistema. Las figuras que pueden aparecer en este grafo son las mostradas en la figura 29.

Aquí encontramos tres tipos de contextos navegacionales: el formado solo por nodos (icono 1), el formado por nodos y otros contextos (icono 2) y los contextos navegacionales que dependen de una entrada del usuario (icono 3).

En estos contextos navegacionales puede haber índices (icono 4), que serían diccionarios, frames, etc.; rutas guiadas (icono 5), que serían páginas que se ejecutan sin necesidad de ser guiadas por el usuario; selecciones (icono 6), que se corresponderían con páginas en las que se espera una respuesta del usuario; menús(icono 6), que se traducirían a páginas o partes de páginas que recogen una serie de opciones para el usuario en forma de menús; nodos externos, que representan enlaces a otras páginas; y por último nodos, que son los contenedores de información.

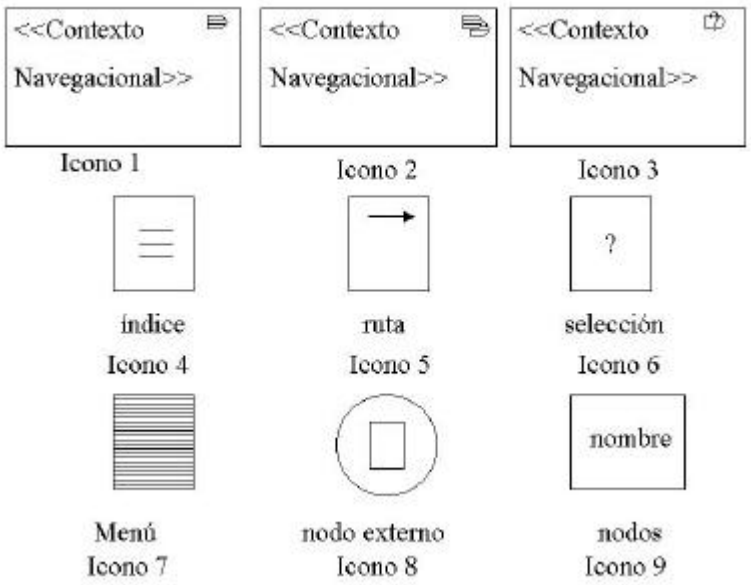

FIGURA 29: ESTEREOTIPOS PARA LOS DIAGRAMAS DE CONTEXTO

Así, para nuestro ejemplo podríamos tener el contexto de la figura 30. Imaginemos que éste es el contexto para un usuario que ha entrado como Arqueólogo pero que también está recogido en el sistema como Etnólogo. Aquí representamos el contexto como Arqueólogo.

Como vemos, en el contexto aparecería un menú: BienMueble. Desde éste, el usuario podría dirigirse al contexto de etnólogo o bien ir a consultar los nodos de identificación de un determinado bien mueble en el nodo de DatosIdentificación. Desde este nodo, podemos ver los datos de descripción del bien en el nodo DatosDescripción.

Como se ve por el sentido de las flechas, desde el nodo de DatosDescripción se permite ir al nodo DatosIdentificación de nuevo. Para volver al menú principal existe un camino desde el nodo DatosIdentificación.

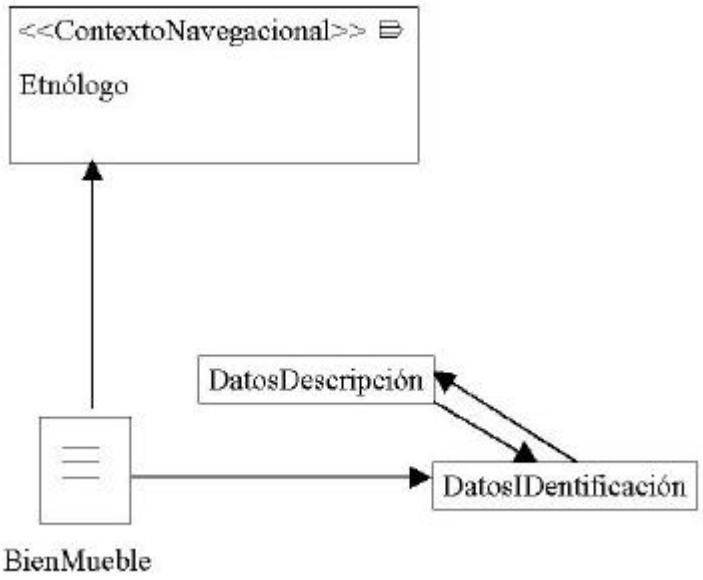

FIGURA 30: EJEMPLO DE DIAGRAMA DE CONTEXTO

?? Diseño de la presentación

El diseño de la presentación soporta el modelo de interfaz para un determinado usuario. Muestra cómo se presenta al usuario la estructura de navegación. Para ello, se hace uso de dos modelos, uno que representa la parte estática de la interfaz y otro que representa la parte dinámica. Estos modelos se definen basándose en el modelo navegacional de tal forma que un mismo modelo navegacional puede dar origen a varios modelos de presentación.

- **Modelo de presentación estático**: Nos va a servir para representar la estructura estática de la interfaz. Para cada usuario se representan de forma abstracta, mediante estereotipos predefinidos, la estructura de las pantallas. Estos estereotipos predefinidos son los mostrados en la figura 31.
  - Anchor: representa a un enlace desde el que se puede navegar.
  - Text: representa un área de texto.
  - Button: representa una zona sobre la que, al hacer click, se ejecuta una aplicación.
  - Form: área en la que se espera que el usuario introduzca datos.
  - Images, audio y video: representa los objetos multimedia de imágenes y gráficos, audio y vídeo respectivamente.
  - Collections: representa a un conjunto de valores posibles. Son la representación de los controles de listas desplegables (ComboBox) y de listas predefinidas (ListBox).
  - Anchored collections: son listas de enlaces.

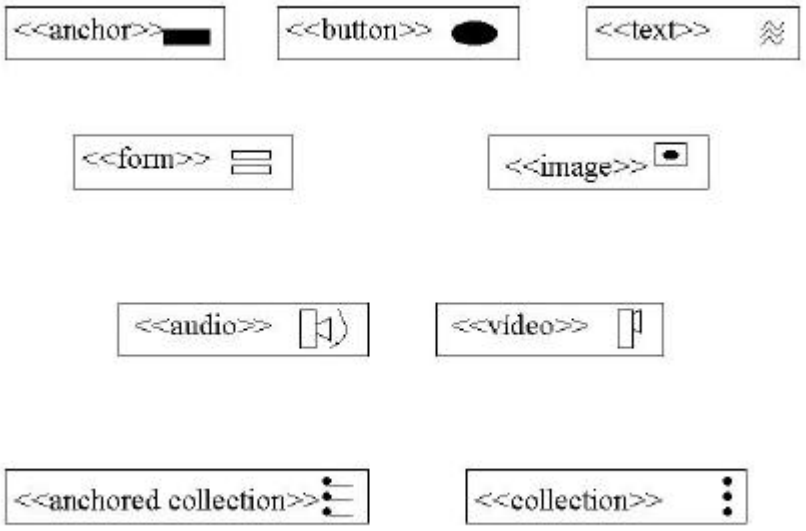

FIGURA 31: ESTEREOTIPOS PARA LA INTERFAZ

- **Escenarios**: para la representación del dinamismo de la interfaz se utilizan los diagramas de estado propuestos en [Booch 1999]. Al ser esta una técnica de UML no va a ser explicada, pues se supone que las técnicas del lenguaje unificado son conocidas. La propuesta de Nora Koch no añade nada nuevo a la definición de Booch, por lo que el lector interesado puede dirigirse al manual de UML.

3. Conclusiones

Las propuestas analizadas en los puntos 5.3.13. y 5.3.14. añaden sin duda nueva semántica y mayor potencia representativa a UML, acercándose bastante a las necesidades que aparecen en el desarrollo de los sistemas de información web.

Sin embargo, ambas propuestas están demasiado cercanas al diseño y a la implementación y dejan al margen flujos tan importantes como la captura de requisitos y el análisis, presuponiendo que los sistemas de información web pueden tratarse en estas etapas al igual que los sistemas clásicos.

Estas propuestas no tienen en cuenta la necesidad de la participación del usuario en el proceso de desarrollo. En los sistemas de información web, la opinión del cliente y los usuarios con respecto a la interfaz es esencial para el éxito del problema. Además, al representar el modelo de presentación estática a tan bajo nivel (casi similar a la implementación) hace que sea más fácil de entender la propia página que los modelos que la definen. Imaginemos una página con un frame en el que hay un menú, en el que hay imágenes, texto, sonidos, videos y colecciones. No es algo poco común y sin embargo la representación de ésta con los iconos de la figura 31 sería muy complicada.

5.3.15. A UML-Based Methodology for Hypermedia Design

1. Conceptos básicos de *the UML-Based Methodology for Hypermedia Design*

Dentro del entorno investigador de los autores anteriores, ha surgido más recientemente un nuevo trabajo que, basado en los anteriores, propone una metodología mucho más elaborada para el desarrollo de aplicaciones hipermedia [Hennicker 2001].

Esta propuesta viene de manos de Rolf Hennicker y Nora Koch y habla de aplicaciones hipermedia en un sentido más amplio del que normalmente se utiliza. Cuando en el entorno de este trabajo se habla de hipermedia, se habla de aplicaciones que en el ámbito de internet trabajan con elementos multimedia e hiperenlaces, así como con importantes almacenes de datos. Son al fin y al cabo lo que se ha denominado en nuestro trabajo sistemas de información web.

En el trabajo se comienza haciendo una análisis comparativo entre el desarrollo de sistemas hipermedia y del resto de sistemas, resaltando entre ellos varias ideas:

- En los sistemas hipermedia existe una mayor diversidad de personas trabajando en el desarrollo de la aplicación: autores, diseñadores, programadores, expertos multimedia, especialistas en marketing, etc. Esto hace más compleja la comunicación entre ellos y la necesidad de que los modelos sean más sencillos en su representación para facilitar esta comunicación.
- En los sistemas hipermedia aparece el concepto de hiperenlace, y con él aparece asociado el concepto de hiperespacio, o lo que es lo mismo la posibilidad de que desde nuestro sistema se puede navegar hacia otros sistemas. Esto provoca que hay que tener controlado este hiperespacio para evitar que el usuario final se pierda.
- Los aspectos cognitivos y estéticos deben ser tratados con la misma importancia en los entornos hipermedia.
- El mantenimiento es mucho más complejo en los sistemas hipermedia.
- La seguridad en los sistemas multimedia debe ser tratada como un aspecto mucho más crítico.

El trabajo de esta propuesta está basado en los trabajos de Conallen (vista en el apartado 5.3.13.) y en los trabajos anteriores de sus propios autores (estudiados en el apartado 5.3.14.). De esta forma, esta propuesta se centra en tres aspectos:

- El contenido, representado mediante el diseño conceptual.
- La estructura de navegación, representada mediante el diseño navegacional.
- El modelo de presentación, representado mediante el diseño de la presentación.

2. Fases de *the UML-Based Methodology for Hypermedia Design*

Para capturar y trabajar con estos tres aspectos la propuesta de Hennicker y Koch propone 4 pasos o fases.

Fase 1- Desarrollo de los casos de uso y el modelo conceptual

Como idea principal, esta propuesta parte de la técnica de Jacobson de los casos de uso, que también utiliza UML. Enuncia que se deben elaborar los casos de uso de la misma forma que propone UML y el Proceso Unificado. La calidad de la elaboración de los casos de uso será crucial, pues de ellos va a nacer toda la elaboración del sistema.

Partiendo pues de estos casos de uso, se debe desarrollar el modelo conceptual del sistema. El concepto del modelo conceptual es equivalente al que se enunció en apartados anteriores. Para conseguir este modelo conceptual, los autores proponen los siguientes pasos:

1. Encontrar las clases
2. Especificar los atributos y operaciones más relevantes
3. Determinar las asociaciones
4. Definir la herencia
5. Encontrar dependencias
6. Identificar interfaces
7. Definir las restricciones

Una vez que el modelo de clases que representa la estructura estática del sistema, el modelo conceptual, es definido, es necesario trabajar con el aspecto de la navegación. Para ello, se divide la navegación en dos ámbitos: el modelo de espacio de navegación y el modelo de la estructura de la navegación.

Fase 2- Construir el modelo de espacio de la navegación

El modelo de espacio de navegación indica qué objetos pueden ser visitados en la navegación. Es una idea equivalente a la del modelo navegacional que se introdujo en el apartado 5.3.14. En el desarrollo de este modelo se toman ideas de diseño que son cruciales y en las que se va a decidir qué vista del modelo conceptual se verá en la navegación.

Los elementos de modelo que van a aparecer en esta fase son las clases navegacionales y las asociaciones entre ellas, que representan la posibilidad de navegación entre dos clases. El resultado a obtener es un modelo de clases para representar la navegación similar al visto en el apartado 5.3.14.

Para llegar al modelo navegacional, se dan tres guías o pautas a seguir:

1. Las clases del modelo conceptual que son relevantes para la navegación se incluyen como clases en el modelo navegacional.
2. Las clases del modelo navegacional pueden contener los mismos atributos que las clases del modelo conceptual, pueden tener atributos derivados añadidos o incluso menos atributos, cuando estos no tienen interés para la navegación.
3. Cuando se pueda navegar entre dos clases navegacionales, debe existir en el modelo navegacional una asociación entre ellas. Por eso es probable que el modelo navegacional añada asociaciones al modelo conceptual.

Fase 3- Construir el modelo de estructura de navegación

El modelo de estructura de navegación va a representar cómo se va a navegar a través del modelo navegacional. Sigue la idea del modelo de contextos navegacionales vistos en el apartado 5.3.14. Sin embargo, en esta propuesta se baja a un nivel más bajo de diseño y se utilizan los estereotipos vistos en la figura 31 para representar la estructura de la navegación.

De esta forma, la estructura de navegación vendrá representada por un grafo dirigido en el que los nodos son índices, rutas, selecciones, menús o nodos.

Para conseguir este grafo, se proponen igualmente una serie de pautas:

1. Para detectar los índices, rutas y selecciones se debe:
   1.1. Considerar sólo aquellas asociaciones del modelo navegacional que tienen una cardinalidad mayor a 1.
   1.2. Cada asociación de este tipo, llevará asociado una o más estructuras de acceso de tipo índices, rutas o selecciones.
2. Para detectar la necesidad de menús:
   2.1. Considerar las clases navegacionales con más de una asociación como fuente
   2.2. Asociar a estas clases un menú.
   2.3. En el menú habrá tantas entradas como roles pueda tener la clase en las asociaciones.

Puesto que esta propuesta está basada en la anterior, el resultado de aplicar este algoritmo a nuestro ejemplo daría como resultado el diagrama que se mostró en la figura 30.

Fase 4- Construir el modelo de presentación

Partiendo de los modelos anteriores, es necesario ya plantearse la estructura abstracta que tendrá la interfaz del sistema. En este paso, los autores de la propuesta dejan muy claro que no hay que representar qué se va a ver en la pantalla, sino la estructura de lo que se va a ver. No hay que entrar pues en detalles de la apariencia física como los colores o los tipos de letras.

Para construir el modelo de presentación, Hennicker y Koch proponen que hay que decidir qué elementos de presentación se van a usar para representar las instancias de las clases navegacionales y los elementos de la estructura de navegación detectados en las pasos 2 y 3. Para ello, usan los estereotipos de Conallen presentados en el apartado 5.3.13.

Como en casos anteriores, se ofrecen las pautas a seguir para conseguir este modelo:

1. Para cada clase navegacional es necesario construir una presentación, así como para cada estructura de navegación.
2. Elegir una estructura de navegación como raíz.
3. Representar las asociaciones de las clases navegacionales con elementos de navegación dentro de las representaciones hechas en el paso 1.

3. Conclusiones

El trabajo que acabamos de analizar es mucho más refinado que los anteriores. A pesar de no aportar nuevas ideas, puesto que se basa en los trabajos vistos en los apartados 5.3.13. y 5.3.14, sí que se puede destacar por varios aspectos. Por un lado, propone una secuencia de pasos y de guías a la hora de obtener los modelos. Esto lo acerca mucho más a lo que sería una propuesta metodológica. Por otro lado, incluye aspectos más alejados del diseño como es utilizar los casos de uso para capturar la funcionalidad del sistema. Sin embargo, no indica cómo conseguir que esos casos de uso capturen los conceptos necesarios para definir la navegación o la interfaz. Este es precisamente uno de los aspectos que se le puede criticar.

Simplemente por el hecho de que se basa en las propuestas anteriores, se le pueden hacer las mismas críticas con respecto a la complejidad de sus modelos y al hecho de que no tiene en cuenta al usuario en el desarrollo de la interfaz.

Por último, decir que esta propuesta está orientada totalmente al proceso. No indica en ningún momento qué hay que presentar al usuario o cuáles son los resultados finales del trabajo. Sólo indica los modelos a conseguir.

5.4.Otras propuestas

Existen actualmente nuevas tendencias y nuevas propuestas en el entorno de la multimedia y de la web. Sin embargo, recoger todas en este documento sería algo inviable. Dentro de todas estas propuestas se han estudiado en los apartado anteriores las más relevantes. Sin embargo no podemos cerrar el capítulo sin mencionar otras tendencias que están naciendo y que están teniendo mucha difusión. El hecho de que estas no hayan sido estudiadas en profundidad es porque no aportan ideas nuevas sobre las anteriores o porque no hay demasiado publicado sobre las mismas.

Así por ejemplo nos encontramos con WebML (Web Modeling Language) [Ceri 2001]. Esta propuesta viene de manos de Stefano Ceri, Piero Fraternali y Aldo Bongio de la Politécnica de Milano. Esta propuesta es interesante sobretodo porque lleva asociado una herramienta que permite implementar bajo XML todo lo que se desarrolla.

Otro trabajo importante es el realizado por Paolo Atzeni y Paolo Merialdo de la Universidad de Roma y por Giasalvatores Mecca de la universidad de la Basilicata [Mecca 1999]. Este trabajo es el proyecto Araneus. Araneus es un modelo de datos específico para describir esquemas de aplicaciones con hipertextos. Está orientado en el mundo de las bases de datos y el modelo entidad relación.

Otra propuesta de interés es OSM. Ésta nace de manos de Liddle, Embley y Woodfiel [Liddle 2001a]. OSM no es en sí una propuesta metodológica, es un modelo orientado a objetos que pretende ser lo suficientemente sólido como para dar soporte a todas las fases del ciclo de vida de un proyecto de desarrollo software (especificación, análisis, diseño, implementación y evolución). Está compuesto a su vez de tres submodelos:

- Modelo de objetos y relaciones entre ellos
- Modelo de comportamiento
- Modelo de presentación

Los componentes estructurales de OSM son objetos y relaciones entre ellos. Los objetos se relacionan mediante dos abstracciones: la generalización/especialización y la clasificación. Un objeto tiene una identidad única y es activo. Además puede actuar de forma concurrente con otros objetos. Las relaciones por su parte se agrupan en conjuntos de relaciones y pueden ser tratadas como objetos dándoles todas las propiedades que se podrían dar a éstos.

El comportamiento viene represetando en OSM usando un diagrama de estados. Cada objeto tiene asociado un conjunto de estados. Para cambiar de estado se deben producir eventos que provoquen la transición de un estado a otro. Por último, la relación con el exterior es recogida mediante objetos de interacción. Un objeto puede sincronizarse y comunicarse con otros objetos mediante interacciones. En [Liddle 2001b] se puede encontrar un ejemplo de modelado de un sistema usando OSM. En realidad toma idea de varios modelos (modelo de clases, diagramas de estados, etc.) y propone sus nuevas formas de representar la estructura, el comportamiento y la interoperabilidad de los sistemas

Pero lo más importante de OSM es que se puede traducir directamente a un lenguaje formal, OSM-Logic compuesto por fórmulas que sirven para representar a esos sistemas. Un importante beneficio de esta formalización es que se puede definir un modelo ejecutable a partir de un modelo OSM usando un razonamiento formal y lógico. Los autores de OSM presentan además una herramienta que nos permite conseguir este modelo ejecutable.

5.5. Estudio comparativo

En este apartado se van a realizar una serie de comparaciones entre las propuestas estudiadas. Estas comparaciones nos servirán para analizar el grado de adecuación de estas metodologías a la hora de desarrollar un sistema de información global.

Tomando las clásicas fases de todo proceso de desarrollo: especificación, análisis, diseño, codificación, pruebas y mantenimiento, la primera evaluación que vamos a realizar a las metodologías enunciadas nos indica cuáles de estas fases genéricas son tratadas en cada propuesta y cuáles de ellas pueden sernos útiles.

Comenzando por la especificación de requisitos. La captura de requisitos es una fase que pocas propuestas tratan, solo HFPM, SOHDM y el Proceso Unificado hablan de ellas. Otras propuestas como OOHDM o EORM asumen que las propuestas de metodologías como OMT serían adecuadas para la captura de requisitos. Dentro de las técnicas que se ofrecen para realizar la captura de requisitos se ofrecen dos: los diagramas de casos de uso y los escenarios. Ambas resultan muy adecuadas para recoger las necesidades funcionales del sistema, pero no permiten especificar qué desea ver el usuario o cómo desea verlo. Como se ha venido diciendo, en los sistemas de información global la comunicación con el usuario es esencial para conseguir un buen resultado y las técnicas de casos de uso y los escenarios no resultan suficientes. Así, tras este estudio podemos concluir que es necesario establecer una propuesta para realizar la captura de requisitos que ofrezca la posibilidad de tener una comunicación fluida con el usuario para establecer correctamente las necesidades y objetivos del sistema.

A la hora de plantear la comunicación y el estudio del entorno donde va a ser implantado el sistema, parece conveniente estudiar las propuestas que hace RNA. Si recordamos, esta propuesta no ofrecía mecanismos de diseño suficientes para el

desarrollo de todo el proceso y estaba orientada exclusivamente a realizar la especificación y el análisis de aplicaciones jurídicas en la web, pero estaba orientaba el desarrollo del proyecto al usuario. Sus ideas de estudiar el entorno de trabajo y las necesidades del sistema podría ser adecuada en una propuesta para el desarrollo de sistemas de información web.

Si pasamos ahora a la fase de análisis, sí que encontramos más propuestas que recogen la necesidad de abordar esta fase de una manera más concreta. Es necesario que en esta fase se analicen y modelen todos los aspectos que van a ser críticos en los sistemas de información global. De las propuestas que encontramos, RNA y WSDM centran el análisis en el estudio de los grupos de usuarios, mientras que las otras se centran en realizar un modelo de clases a alto nivel para representar el modelo conceptual del sistema. Las dos ideas resultan adecuadas, pero ninguna es suficiente por sí misma. Los sistemas de información global pueden variar sustancialmente, sobretodo en su interfaz y en las posibilidades de trabajo, dependiendo de qué usuario sea el que trabaje en ese momento con él. También suelen ser sistemas que tienen una necesidad de almacenamiento interno de información muy compleja, por lo que es necesario estudiar y representar esta complejidad en las primeras fases del ciclo de vida y parece que el modelo de clases resulta una técnica adecuada para ello. Es necesario pues que en el análisis de un sistema de información web se estudie tanto el modelo conceptual como la variabilidad del sistema en función de los usuarios. Pero tampoco hay que dejar a un lado el estudio de la funcionalidad o de la interfaz, de la navegación o de la multimedia. Es por tanto necesario el realizar una propuesta capaz de analizar todos estos aspectos.

La fase de diseño es la que se encuentra más desarrollada en todas estas propuestas. Todas ellas, excepto RMM y HDM, asumen el paradigma de la orientación a objetos, puesto que se ajusta más a las características de estos sistemas. Otra idea que es asumida por la mayoría de las propuestas es la necesidad de diferenciar en diseño la representación conceptual de los aspectos de navegación e interfaz.

Planteemos el diseño conceptual. La gran mayoría de las propuestas utilizan los diagramas de clase para representar las necesidades de almacenamiento básicas del sistema, técnica que ha dado en todas ellas muy buenos resultados y que es adecuada asumir gracias también a lo estandarizada que está esta técnica.

Con respecto al diseño navegacional, hay que diferenciar dos partes. Por un lado, la mayoría de las propuestas asumen la idea original de OOHDM y EORM de representar la navegación mediante un diagrama de clases. Sin embargo, cuando esta navegación varía en función del contexto en el que trabajemos, es necesario diferenciar cada uno de estos contextos. Las posibilidades de representar estos contextos son múltiples: los contextos navegacionales de OOHDM, las propuestas de Conallen y de Koch, etc. Sin embargo, todas estas resultan muy complejas en sistemas donde la navegación sea complicada. Es necesario, pues buscar alternativas de representación.

Con respecto a la interfaz, casi todas las propuestas hacen hincapié en el uso de técnicas que representen la interfaz de forma abstracta y ven necesario la necesidad de representar tanto la estructura de las pantallas como su dinamismo. Sin embargo, estos modelos no parecen adecuados en su mayoría. Algunos de ellos, como los propuestos por Conallen y Koch son demasiados cercanos a la implementación y en muchos casos, sobretodo cuando las aplicaciones son complejas, es más fácil presentar la pantalla diseñada que el modelo. Otras propuestas como los ADV de OOHDM son demasiado ambiguas y no se indica muy bien el nivel de abstracción al que se debe llegar. Por ello,

es necesario normalizar la representación y marcar unas pautas que el diseñador pueda seguir para conseguir los prototipos.

Algo que también se echa en falta en todas las propuestas es el plantear como necesario el estudio de la arquitectura del sistema o su división modular. Esto es debido a que las aplicaciones multimedia y de la web, que son las que ocupan a la mayoría de estas propuestas, son sencillas y no es necesario realizar este estudio. En los sistemas de información global es necesario plantearse estas actividades como esenciales en el diseño.

Algo que también parece adecuado es el hacer uso de los patrones de diseño, que tanto se están usando actualmente. Sólo MacWeb hace referencia a esta técnica. El hacer uso de estos patrones puede llevar a la reutilización y tener un mantenimiento menos costoso del sistema.

Con respecto a la implementación, son pocas las propuestas que no la referencia. Sin embargo, apenas ninguna de ella propone técnicas de implementación. Quizás las propuestas de Conallen y Koch se acercan un poco a esto pues sus modelos son tan cercanos a la implementación que es fácil obtener el código. También es interesante el trabajo de OO-Method y la herramienta que ofrece para trabajar con ella, que se encarga de generar el código.

Las fases de prueba y mantenimiento son enunciadas en algunas propuestas, pero en ninguna de ellas se ofrecen técnicas o métodos a aplicar a la hora de realizar estas fases.

Como resumen a esta idea, en la tabla 7 se han recogido todas las propuestas analizando en qué fases se centran cada una de ellas.

Metodología	Especific.	Análisis	Diseño	Codificac.	Pruebas	Mantenim.
HDM			X			
RMM			X	X		
EORM			X	X		
MacWeb			X	X	X	
OOHDM			X	X		
WSDM		X	X	X		
OO-Method			X	X	X	X
SOHDM	X	X	X	X	X	
RNA		X			X	
HFPM	X	X	X	X	X	X
OO/Pattern		X	X	X	X	
Proceso Unificado	X	X	X	X	X	
Building Web Applications with UML			X			
Specification and modeling of multimedia and hypermedia systems			X			
A UML-Based Methodology for Hypermedia Design		X	X			

TABLA 7: FASES RECOGIDAS EN CADA PROPUESTA

También se presenta en la tabla 8 un resumen de las técnicas propuestas y se indica qué metodologías la aplican.

	HDM	RMM	EORM	MacWeb	OOHDM	WSDM
ERD	X	X				
Modelo de clases			X	X	X	X
Clases navegacionales: nodos, enlaces, etc.			X	X	X	
Escenarios						
Técnicas GUI		X	X	X	X	X
Patrones de diseño				X		
Casos de uso						
Técnicas y modelos propios		X			X	

	OO-Method	SOHDM	RNA	HFPM	OO/Patterns
ERD					
Modelo de clases	X	X		X	X
Clases navegacionales: nodos, enlaces, etc.	X	X		X	X
Escenarios		X			
Técnicas GUI			X	X	X
Patrones de diseño				X	X
Casos de uso					X
Técnicas y modelos propios	X				

	Proceso Unificado	Building web applications with UML	Specification an modeling of multiemedia and hypermedia systems	A UML-Based Methodology for Hypermedia design
ERD				
Modelo de clases	X	X	X	X
Clases navegacionales: nodos, enlaces, etc.		X	X	X
Escenarios	X			X
Técnicas GUI	X	X	X	X
Patrones de diseño			X	X
Casos de uso	X			X
Técnicas y modelos propios	X			X

TABLA 8: TÉCNICAS Y MODELOS USADOS EN LAS METODOLOGÍAS

Puede verse que en las primeras propuestas la tendencia era de basarse en los ERD y que la tendencia ha evolucionado hacía el paradigma de la orientación a objetos. Vemos que en las últimas propuestas el uso de técnicas de captura de requisitos, como el uso de

escenarios o de casos de uso, se ha ido implantando. Esto ha sido debido a que, a medida que estas técnicas han evolucionado a sistemas más complejos, la necesidad de disponer de un mecanismo de definición de requisitos ha ido en aumento.

Vemos también que hay bastantes modelos que proponen técnicas y modelos propios, aunque si analizamos el recorrido por todas las propuestas que hemos visto en el apartado anterior, veremos que son bastante similares.

Por último, podríamos resaltar algunos aspectos importantes que parecen adecuados para el desarrollo de los sistemas de información global. Estos aspectos se podrían separar en:

a) Una propuesta para sistemas de información global debe ofrecer mecanismos para recoger la variabilidad de la aplicación dependiendo del actor que lo utilice. En la primera columna de la siguiente tabla, vemos que sólo WSDM recoge este aspecto.

b) En los sistemas de información global, para asegurar la reutilización y facilitar el mantenimiento, es adecuado separar los conceptos de diseño básico o diseño conceptual del diseño navegacional. Vemos en la segunda columna de la tabla 9 que este aspecto está recogido en más propuestas.

c) Por la misma razón que en el apartado b, también resulta adecuado separar la navegación de la interfaz. En la tercera columna de la tabla 9, vemos que no todas las propuestas ofrecen esta posibilidad.

d) Además, una propuesta para sistemas de información global debe ofrecer mecanismos para trabajar adecuadamente con la multimedia. Como se ve en la tabla 9, sólo las metodologías multimedia trabajan este aspecto.

Metodología	Variabilidad usuario	Separación Conceptual-Navegación	Separación Navegación-Interfaz	Multimedia
HDM		X		X
RMM		X	X	
EORM		X		X
MacWeb		X	X	
OOHDM		X	X	X
WSDM	X	X		
OO-Method		X	X	
SOHDM		X	X	X
RNA		X		
HFPM		X	X	
OO/Pattern		X		
Proceso Unificado		X		
Building Web Applications with UML				X
Specification and modeling of multimedia and hypermedia systems		X	X	
A UML-Based Methodology for Hypermedia Design		X	X	

TABLA 9: RESUMEN DE ASPECTOS

Para resaltar las ideas finales podemos indicar que:

?? La gran mayoría de las propuestas usan técnicas de modelado orientado a objetos, algo bastante lógico pues la orientación a objetos resulta un paradigma bastante adecuado para el desarrollo de estas aplicaciones.

?? La gran mayoría de las propuestas se centran principalmente en diseño, la mayoría supone el análisis realizado o propone los casos de uso o los escenarios para realizarlos.

?? Un número muy alto de propuestas, indican que es necesaria la separación de la navegación y el diseño de la interfaz del diseño propiamente dicho.

?? Ninguna de ellas ofrece un marco de referencia para trabajar con los sistemas de información global, puesto que no cubren completamente el trabajo de todos los aspectos que caracterizan a este sistema.

6. Proyecto de investigación

6.1. Introducción

Una vez vistas las tendencias actuales, se pueden concluir muchos aspectos. Por un lado, podemos observar la gran cantidad de modelos nuevos que aparecen moviéndose en torno a la misma problemática, sin que haya unanimidad ni siquiera en la definición de los términos.

Los autores que se han estudiado coinciden sin embargo en algunos aspectos a destacar:

- La gran mayoría de las propuestas aceptan la orientación a objetos como un paradigma adecuado para el desarrollo de las aplicaciones en la web. Y dentro de ella, la nomenclatura de UML parece la más adecuada debido a que es la más ampliamente aceptada.
- Normalmente todos apoyan la necesidad de incluir aspectos de navegación e interfaz y la mayoría apuesta porque estos aspectos se trabajen de forma independiente. Esto da una mayor independencia entre los dos aspectos y permite mayor reusabilidad y mayor facilidad a la hora del mantenimiento tanto adaptativo como aumentativo.
- Todas asumen que los sistemas de desarrollo en la web tienen ciertos aspectos como son la multiplicidad de medios con los que se trabaja, el hiperespacio en el que se trabaja, la importancia de la interfaz adecuada e intuitiva, etc.; que hacen que estos sistemas sean más complejos que los sistemas clásicos por lo que es necesario la definición de nuevos modelos y nuevas técnicas que permitan capturar estas características.

Sin embargo, se pueden encontrar algunos puntos que no quedan muy claros dentro de todas estas propuestas. El primero de ellos es que todos los autores en sus trabajos indican que en los sistemas de información la comunicación con los clientes debe ser más fluida y éste debe participar de forma más activa en el desarrollo del proyecto. Los sistemas de información global son sistemas complejos en los que su calidad se mide por la información que almacena y la presentación que hace de ésta. Por tanto, es necesario contar con la ayuda de un experto en la materia, el cliente, que guíe a los desarrolladores en el proceso [González 2001].

Pues bien, a pesar de que todos los autores asumen esta idea, la gran mayoría deja a un lado todos los aspectos de captura de requisitos. Algunos proponen técnicas como los casos de uso o los escenarios, pero sin aportar nada nuevo con respecto al uso de estas técnicas en este entorno y asumen que se aplican de la misma forma que en los sistemas clásicos. Al utilizar estas técnicas se echa en falta el apoyo de guías que nos ayuden a capturar la información que requerimos del usuario para tener un conocimiento adecuado del problema.

Otro aspecto que se detecta es que hay metodologías que proponen un proceso de trabajo, más o menos completo, como HFPM y otras que proponen modelos para la representación del sistema, como OO-Method u OOHDM, pero ninguna de ellas ofrece una guía que indique qué hay que hacer en cada fase del proceso del desarrollo, qué hay que obtener y qué técnicas se pueden o se deben aplicar para ello.

El trabajo que se propone realizar en este proyecto de investigación consiste precisamente en eso. Tomando las buenas ideas ya estudiadas por las propuestas que hemos ido analizando, vamos a definir un marco metodológico para sistemas de información global en el que se va a cubrir por completo el ciclo de vida del proyecto y en el que, para cada fase, el desarrollador va a tener una guía de cómo hacer las cosas, qué presentar al cliente y qué obtener después de cada fase.

La propuesta en la que se pretende trabajar se fundamenta en la orientación a objetos y, tomando como base los razonamientos de los autores vistos hasta ahora, trata de forma especial los aspectos multimedia, de navegación y de interfaz, sin dejar a un lado los aspectos más clásicos de los sistemas software como son la funcionalidad o las necesidades de almacenamiento de información.

A continuación se va a presentar una visión genérica de dicha propuesta en la se está trabajando actualmente. Tras esto se hará una evaluación de la misma y se presentará el estado en el que se encuentra la elaboración concreta de ésta. El trabajo de investigación está actualmente elaborándose, por lo que se encontrarán más desarrollados unos aspectos que otros. El proceso de trabajo que se está siguiendo en el desarrollo de la propuesta se basa en que una vez realizada la definición del ciclo de vida, se vayan estudiando los flujos de trabajo, de manera que para cada uno de ellos se analicen y proponga el proceso a seguir teniendo presente la experiencia de otros grupos de trabajo en el sector. Tras esto se hace una propuesta del producto a obtener y por último se hace una prueba real dentro del marco empresarial. Actualmente se está terminando el flujo de especificación y el flujo de análisis está en sus primeras fases. Los otros flujos de trabajo son meramente una propuesta que aún no ha sido pulida y que simplemente se introducen.

6.2. Bases de la propuesta

Una vez analizada todas las propuestas, se han obtenido una serie de conclusiones que son ideas generales de todas ellas y que vamos a asumir en nuestra propuesta. El éxito que estas propuestas han tenido a la hora de utilizar estas ideas, avalan la decisión de tomarlas como adecuadas para nuestra propuesta. Por ello, nuestra metodología va a ser una metodología:

- Orientada a objetos,
- Que separa el aspecto de navegación del diseño básico,
- Que separa la interfaz de la navegación.

Nuestra propuesta debe además tener una serie de características añadidas:

- Debe cubrir todo el ciclo de vida del proyecto.
- Debe estar orientada al proceso, es decir, debe indicar qué hacer en cada momento del ciclo de vida.
- Debe estar orientada al producto, es decir, en cada fase se indicará qué hay que obtener y el formato que deberá tener para ello.
- Debe ser sencilla, sobretodo en sus primeras fases, para facilitar la participación del cliente y del usuario.
- Debe ser completa, para cubrir todas las necesidades del desarrollador y ofrecer una semántica suficiente como para trabajar de forma adecuada todos los aspectos críticos que se han venido destacando de los sistemas de información global.

Partiendo de estas premisas, en este documento se presenta una visión global de toda la metodología. El objetivo final es ir concretando cada una de las ideas que se indican de forma general en este resumen, hasta conseguir una propuesta completa y adecuada para este entorno.

A continuación se va a presentar esta visión genérica, comenzando por una presentación del ciclo de vida. Tras esto se analizarán cada uno de los flujos de trabajo indicando sus objetivos, las técnicas que se usan y los productos a obtener en cada uno de ellos. Finalmente se hará una valoración general de la propuesta y se comentará el estado de desarrollo actual de la misma.

6.3. Propuesta metodologíca para el desarrollo de sistemas de información global

6.3.1. Ciclo de Vida

Para definir el ciclo de vida de nuestra propuesta, partimos de la idea de dividir la vida del proyecto en flujos de trabajo. El ciclo de vida comprenderá un total de cinco flujos de trabajo: especificación, análisis, diseño, codificación y pruebas. En principio estos flujos de trabajo se realizarán de forma consecutiva, pero si nos basamos en la realidad de los proyectos software, desde un determinado flujo es necesario volver a flujos anteriores para redefinir nuevos aspectos. Por ello, nuestra propuesta será secuencial pero permitirá iterar y volver a flujos anteriores.

En las propuestas que se han presentado, en el flujo de diseño se estudiarán tres aspectos diferentes: el diseño básico de la aplicación, el diseño de la navegación y el diseño de la interfaz. Asumiendo esta idea de estudiar los aspectos de navegación e interfaz de forma independiente, el flujo de diseño se va a dividir en nuestra propuesta en tres actividades: diseño básico, en el que se estudia el diseño conceptual del sistema; el diseño navegacional, en el que se estudia el sistema de navegación del sistema; y el diseño de interfaz abstracta, el que se estudian los aspectos de presentación e interacción con el usuario de forma abstracta.

Esta división que aparece reflejada en muchas propuestas, también se ve reflejada en todos los flujos de trabajo de la metodología que se está presentando, no sólo en diseño. Así , ya desde los primeros flujos de trabajo se hace una separación entre lo que se almacena en el sistema, lo que se puede hacer con el sistema y de qué forma se va a presentar la información al usuario.

Aunque a continuación se van a presentar cada uno de estos flujos, a modo de resumen, en la tabla 10 se indica cuáles son los productos que hay que obtener en cada uno de los flujos de trabajo propuestos.

Flujo de trabajo	Productos resultantes
Especificación de requisitos	* Catálogo de requisitos del sistema que recogerá: - Los objetivos del sistema - Los requisitos de almacenamiento de información - La definición de actores - Los requisitos funcionales - Los requisitos de interacción - Los requisitos no funcionales
Análisis	* Documento de análisis del sistema que recogerá: - El modelo de clases del sistema - El modelo de navegación - Los prototipos de interfaz
Diseño	* Documento de diseño del sistema que recogerá: - La arquitectura abstracta del sistema - La división del sistema en subsistemas - El diseño de los casos de uso - El modelo de clases de diseño - Modelo de clases navegacionales - Contextos navegacionales - Modelo de clases navegacionales refinado con los aspectos de interfaz abstracta - Modelo dinámico de la interfaz abstracta
Implementación	* Programa ejecutable * Manual de usuario
Pruebas	* Plan de pruebas * Resultado del plan de pruebas

TABLA 10: RESUMEN DE LA PROPUESTA METODOLÓGICA Y DE LOS PRODUCTOS RESULTANTES

A continuación vamos a ir analizando cada uno de los flujos de trabajo de forma más concreta. La fase de mantenimiento quedará al margen en este momento. Sería conveniente anotar que, ya que ninguna de las propuestas que hay para el desarrollo de este tipo de sistemas propone técnicas a aplicar en esta fase, habría que recurrir a otras propuestas realizadas para proyectos de desarrollo software, con el fin de encontrar una referencia en la que basarse.

Como ya se comentó, el estado de desarrollo actual de la propuesta no es homogéneo para todos los flujos de trabajo. Como se verá, la especificación de requisitos y el análisis se encuentran mucho más detallados que los flujos siguientes.

Para aclarar las ideas propuestas, vamos a plantearnos como ejemplo el desarrollo de la aplicación de bienes muebles que se introdujo en el apartado 5.2. para estudiar las propuestas presentadas.

6.3.2. Especificación de requisitos

Para estudiar los tipos de requisitos que habría que recolectar en el desarrollo de sistemas de información global, lo primero que habría que tener en cuenta son las características que ya hemos destacado de este tipo de sistemas. Por un lado, es necesario estudiar las necesidades de almacenamiento y funcionalidad, importantes en todos los sistemas de información [Durán 2000] [Robertson 2000], pero además hay que recabar información referente a esos aspectos que acercan a los sistemas de información

global a las aplicaciones hipermedia y de la web, como podrían ser los aspectos de navegación y de interfaz abstracta de usuario.

De esta forma, de la fase de especificación de requisitos se debe conseguir el catálogo de requisitos del sistema que englobe:

- La definición de los objetivos del sistema.
- Los requisitos de almacenamiento de información.
- La descripción de los actores del sistema.
- Los requisitos funcionales, descritos a través de los casos de uso.
- Los requisitos de interacción, en lo que se recogerá el sistema de navegación de la aplicación y la interacción con el usuario.
- Los requisitos no funcionales. Estos son otra serie de requisitos como los requisitos de comunicaciones del sistema, de seguridad, de portabilidad, etc. que en la mayoría de los sistemas es necesario recoger para garantizar su adecuación a las necesidades.

El flujo de definición y captura de requisitos es la que está más avanzado en el desarrollo de la propuesta. Al igual que otras fases está dividida en actividades que se dividen a su vez en tareas. Actualmente estamos trabajando en la definición de estas actividades y tareas.

Las actividades y tareas que engloban este flujo de trabajo son:

- Actividad 1: Obtener información sobre el problema y determinar objetivos
  - Tarea 1.1- Obtener información sobre el dominio del sistema
  - Tarea 1.2- Preparar y realizar las entrevistas y reuniones
  - Tarea 1.3- Identificar los objetivos del sistema
- Actividad 2: Identificar y definir los requisitos de almacenamiento de información
  - Tarea 2.1.-Identificar y definir los requisitos de almacenamiento de información
  - Tarea 2.2.- Describir la naturaleza de los datos
- Actividad 3: Estudiar a los actores
  - Tarea 3.1- Definir los actores básicos del sistema
  - Tarea 3.2- Definir la generalización de los actores
  - Tarea 3.3- Definir la incompatibilidad de actores
  - Tarea 3.4- Definir los actores derivados
- Actividad 4: Identificar y definir los requisitos funcionales del sistema
  - Tarea 4.1.- Definir los diagramas de casos de uso
  - Tarea 4.2- Describir los casos de uso
- Actividad 5: Identificar y definir los requisitos de interacción
  - Tarea 5.1- Identificar y definir los criterios de recuperación

- Tarea 5.2- Identificar y definir los prototipos de visualización

✍ Actividad 6: Identificar y definir los requisitos no funcionales del sistema

- Tarea 6.1.- Definir los requisitos no funcionales

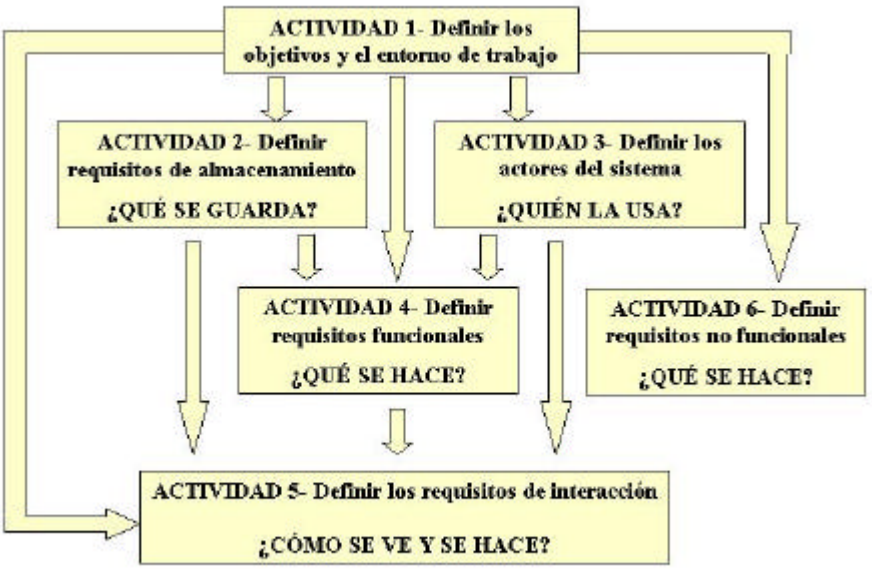

FIGURA 32: RECOLECCIÓN DE REQUISITOS

A modo gráfico, en la figura 32 se presenta una visión global de la estructura de la propuesta para la especificación de requisitos, que se inicia definiendo los objetivos que se pretenden alcanzar con el sistema a desarrollar (actividad 1), a partir de ellos se definirán los requisitos de almacenamiento que el sistema presenta (actividad 2) y los actores que se identifican para interactuar con el sistema (actividad 3). Una vez conocidos los requisitos de almacenamiento e identificados los actores se definirán los requisitos funcionales a los que el sistema debe atender (actividad 4). La expresión de la funcionalidad del sistema se completa con la indicación de a qué información accede cada actor, para qué accede a ella y en qué orden lo hace (actividad 5). Por último, la especificación de requisitos se completa con la definición de otras necesidades que pueden presentar estos sistemas como son los requisitos de seguridad, salvaguarda, rendimiento, etc. (actividad 6). Las flechas en el diagrama indican la repercusión que una actividad tiene en las demás. De esta forma, la definición de objetivos va a influir en la definición de todos los requisitos y la definición de los requisitos de interacción se va a ver afectado por las definiciones realizadas en las otras actividades.

Actividad 1: Obtener información sobre el sistema y determinar objetivos

La primera actividad que hay que realizar es la de definir y conocer el sistema y el entorno de trabajo en el que se va a desarrollar la aplicación. Para ello, es básico obtener información sobre el dominio del sistema, estudiando folletos, otros sistemas, etc. Así como establecer el plan de entrevistas que permita al equipo de trabajo detectar los objetivos y los requisitos de la aplicación.

Por ello, una vez realizadas estas tareas es necesario que se definan los objetivos del sistema. Para ello, se ha analizado la propuesta de Entorno Metodológico de Ingeniería de Requisitos para sistemas de información [Durán 2000], así como Volere [Robertson 2000]. De esta forma se define un patrón que se recogerá en el documento de requisitos en el que se van a almacenar de forma estructurada y bajo un identificador único cada uno de los objetivos del sistema.

Estos objetivos van a servir como base para la definición del resto del sistema, de manera que los requisitos que se identifiquen y definan en las fases siguientes tendrán que hacerse en base a un objetivo concreto.

Para nuestro ejemplo, de los objetivos que se podrían intentar alcanzar con los bienes muebles, podría ser el hecho de que la aplicación permitiese la catalogación y gestión de todos los bienes muebles. Este objetivo se recogería en un patrón similar al que se muestra en la tabla 11.

OBJ-01	Gestionar los bienes muebles
Descripción	El sistema deberá permitir la gestión de los datos de los bienes muebles de Andalucía. Estos datos se dividirán en: - Datos de Identificación - Datos de Descripción
Comentarios	Los bienes muebles pueden ser dados de alta por diferentes usuarios y pueden ser modificados de forma concurrente

TABLA 11: EJEMPLO DE OBJETIVO DEL SISTEMA

En la propuesta final de la metodología, esta tabla es más completa. En ella se podrán introducir otros datos como los autores o los usuarios que han permitido detectar el objetivo. Sin embargo, este documento no pretende presentar la metodología de forma exhaustiva, sino dar una visión global de la misma.

Actividad 2: Identificar y definir los requisitos de almacenamiento de información

Cuando ya se conocen los objetivos de la aplicación es necesario plantearse qué se va a almacenar en el sistema. Es necesario identificar sobre qué conceptos se desea guardar información y cuál va a ser la información concreta almacenada sobre esta.

Por ejemplo, en nuestra aplicación para la gestión de los bienes muebles sería lógico pensar que vamos a almacenar información sobre los bienes muebles. Una vez detectada esta necesidad debemos identificar qué almacenar de forma concreta sobre estos bienes.

Para recoger estos datos, proponemos hacer uso de un nuevo patrón que nace de la idea de la *Metodología de Elicitación de Requisitos* [Durán 2000], pero que se modifica en algunos aspectos. Por ejemplo, para el caso de los bienes muebles podría ser similar a la que se muestra en la tabla 12.

Como se puede ver, se han incluido más atributos al bien mueble de los que se describieron en un inicio. Estos atributos se han añadido para describir con más precisión algunos detalles de la propuesta que se está presentando. El significado de estos atributos se describe en el patrón.

RA-01	Bien Mueble	
Objetivos asociados	OBJ-01 : Gestionar los bienes muebles	
Descripción	El sistema deberá almacenar la información correspondiente a los bienes muebles de las provincias de la Comunidad autónoma. En concreto:	
Datos específicos	**Nombre y Descripción**	**Naturaleza**
	Código del bien: Es el número de identificación que recibe el bien según el sistema de información del patrimonio histórico.	Código
	Denominación: Es el título del bien mueble según su catalogador	Cadena
	Otra denominación: Es otro nombre que puede tener el bien.	Cadena
	Titulado/Representado: Un bien puede denominarse como titulado o como representado, según si tiene un titulo propio o es representado por otro bien.	Enumerado Valores:{Titulado, Representado}
	Ubicación habitual: Es el nombre del lugar (museo, iglesia, etc.) en que el bien se encuentra normalmente ubicado.	RA-02
	Ubicación temporal: Es el nombre del lugar (museo, iglesia, etc.) en el que, por alguna razón especial, se encuentra el bien.	RA-02
	Acceso: Este campo sirve para indicar si el bien es de dominio público o no	Enumerado Valores:{Público, Restringido}
	Cronología: Se compone de un intervalo de años y recoge la cronología del bien, es decir, el período en el que se supone que se hizo.	Cronología
	Certeza: Indica el grado de certero que tiene la cronología.	Enumerado Valores:{Exacta, Incierta, Aproximada}
	Imágenes: Recoge todas las imágenes que se tienen de un bien mueble	RA-10 Cardinalidad: 0..n
	Períodos Históricos: Se refiere a todos los períodos históricos dentro de los que se ubica el bien	RA-04 Cardinalidad: 0..n
	Estilo: Recoge todos los estilos artísticos que se encuentran plasmados en el bien	RA-04 Cardinalidad: 0..n
	Tipologías: Es una lista de todos los tipos de obras dentro de los que se puede ubicar el bien	RA-04 Cardinalidad: 0..n
	Autor: Almacena una lista de todos los autores del bien.	RA-09 Cardinalidad: 0..n
	Observaciones: Cualquier tipo de información añadida al bien	Cadena

TABLA 12: EJEMPLO DE REQUISITO DE ALMACENAMIENTO DE INFORMACIÓN

Como vemos, el requisito se identifica de forma unívoca mediante un código (RA-01). Esto nos va a permitir identificarlo en otros patrones. Además hace referencia al objetivo OBJ-01 que se definió en la tabla 10. Esto indica que el objetivo OBJ-01 se va a alcanzar, si no totalmente, sí en parte al cubrir este requisito. Después se ven todos los datos concretos que se va a almacenar para los bienes muebles. Como se puede observar, en estos datos concretos se recoge su nombre, su descripción y su naturaleza.

La naturaleza se refiere a la tipología de la información. En la metodología se definen una serie de naturalezas básicas, como son las cadenas, los enumerados, las imágenes, los sonidos o los números, entre otros. Pero también puede que la naturaleza de un dato sea otro requisito de almacenamiento de información. Como en el caso de los autores. Esta posibilidad permite relacionar unos requisitos de almacenamiento con otros.

Además, para un campo podemos indicar su cardinalidad. Es decir, si el requisito va a tener más de un valor para un dato concreto determinado.

Cuando los datos concretos no tengan como naturaleza una de las naturalezas básicas u otro requisito de almacenamiento, será necesario definir una nueva naturaleza. Es el caso del dato código del bien de nuestro ejemplo. La definición de las nuevas naturalezas es el objetivo de la tarea 2.2. Nuevamente se hará uso de un patrón para ello. Por ejemplo, en la tabla 13 vemos la definición de la naturaleza código.

Nombre	Código	
Significado	Representa el código oficial que se da a cada uno de los bienes muebles	
Formato	**Campo**	**Naturaleza**
	Provincia: Son dos números que representa la provincia en la que se encuentra el bien codificada según el instituto Nacional de Estadística	Numérico
	Municipio: Son tres números que representa la provincia en la que se encuentra el bien codificada según el instituto Nacional de Estadística	Numérico
	Inmueble: Son cuatro números que representa el código del edificio en el que se ubica el bien	Numérico
	Mueble: Son cuatro números secuenciales que el sistema asigna automáticamente para ir identificado de forma unívoca cada bien mueble	Numérico
Restricciones	Las provincias, municipios e inmuebles deben haber sido catalogados por la organización que gestiona a los bienes previo al alta de un bien mueble	

TABLA 13: EJEMPLO DE DESCRIPCIÓN DE NATURALEZA

Una vez definidos los requisitos de almacenamiento y la naturaleza de los datos, se tienen sentadas las bases para la definición de la estructura conceptual de la aplicación.

Actividad 3: Estudiar a los actores

Una vez definidos los requisitos de almacenamiento de información del sistema hay que identificar los actores capaces de interaccionar con el mismo, siendo éste el objetivo de esta tercera actividad. En sentido amplio un actor es una abstracción de una persona externa, de un proceso o de una cosa que interactúa con el sistema. Cada actor define un rol que los usuarios asumen cuando interactúan con el sistema. En esta tarea se definirá qué roles pueden aparecer, es decir, los actores de la aplicación, pero además se van a analizar las incompatibilidades que presentan y las relaciones de generalización entre ellos.

La primera tarea a realizar es la definir los actores básicos de la aplicación. Un actor básico es todo actor que se puede identificar de forma individual atendiendo a algún tipo de criterio o punto de vista a la hora de interaccionar con el sistema software. La experiencia dice que a la hora de identificar los actores básicos que interactúan con un sistema software, pueden existir diferentes criterios o puntos de vista para hacerlo. La aplicación de cada uno de ellos resultará en la identificación de un grupo determinado de actores básicos. Cada actor básico corresponderá a un rol individualizado de interacción con el sistema software. Para definir estos actores en el documento de requisitos, se hará uso nuevamente de un patrón.

Así, en el sistema para la gestión de los bienes muebles se podrían detectar que los actores básicos del sistema podrían clasificarse según diferentes criterios. De esta forma, un mismo actor puede tomar roles de diferentes grupos. Las clasificaciones que

aparecen en el sistema para nuestro caso son dos, una en base al perfil que pueden desempeñar en el sistema y otra en base al área de investigación que ocupen.

En el primero de ellos nos encontramos tres roles básicos:

- ?? Los usuarios del centro investigador. Son los que más privilegios tienen, pueden dar de alta, modificar o consultar cualquier dato. Además son los encargados de validar la información del sistema.
- ?? Los catalogadores. Estos usuarios van a poder introducir información en el sistema, pero la información que introduzcan no se considerará definitiva hasta que no sea validada por un usuario de la organización investigador. En principio se pueden localizar en cualquier lugar del mundo.
- ?? Los usuarios por internet. Solo podrán realizar consultas sobre los datos.

En la clasificación según el área de investigación nos encontramos a los arqueólogos, los etnólogos y los artistas. La diferencia entre unos y otros son los datos que ven. Por ejemplo, los datos sobre actividades solo interesan a los etnólogos, o los datos sobre iconografías solo interesan a los artistas.

Una vez identificados los actores básicos, habría que definirlos y recogerlos en el documento de especificación de requisitos. Para ello, se utilizará nuevamente un patrón. Así por ejemplo para los arqueólogos podría ser la que se muestra a continuación en la tabla 14.

AC-01	Arqueólogo
Objetivos asociados	OBJ-01: Gestionar los bienes muebles OBJ-05: Gestionar los perfiles investigadores
Clasificación	Este es uno de los posibles roles dentro del sistema cuando hacemos una clasificación de los actores en base al perfil investigador del actor.
Descripción	El sistema deberá prever el tratamiento de los usuarios que pertenecen al grupo descrito como arqueólogo que se refiere a personas que entran en el sistema mediante el perfil de investigador arqueólogo.

TABLA 14: EJEMPLO DE DESCRIPCIÓN DE UN ACTOR BÁSICO

Nuevamente este patrón es sólo una visión general del definido en la propuesta.

Tras esta definición, se puede analizar la incompatibilidad de roles entre dichos actores, de tal forma que dos actores básicos declarados incompatibles entre sí no pueden formar parte como actores componentes de un mismo actor derivado.

Dos actores básicos se dice que son incompatibles cuando sus roles asociados no pueden ser asumidos conjuntamente por ningún actor derivado. La incompatibilidad se puede presentar entre actores básicos del mismo grupo (mismo criterio de clasificación) o de grupos diferentes (diferentes criterios de clasificación).

Para definir la incompatibilidad entre actores se utilizará una tabla de doble entrada como la mostrada en la tabla 15.

Cada fila y cada columna corresponde a un actor básico, de manera que en cada fila se indicará las incompatibilidades que presente el actor correspondiente.

Actores básicos	Usuario del centro	Catalogador	Usuario internet	Arqueólogo	Artista	Etnólogo
Usuario del centro	-	X	X			
Catalogador	X	-	X			
Usuario internet	X	X	-			
Arqueólogo				-		
Artista					-	
Etnólogo						-

TABLA 15: DEFINICIÓN DE LA DISJUNCIÓN ENTRE ACTORES BÁSICOS

La disposición de actores por fila y columna se ordenará de igual forma en ambos casos, reflejando una estructura plana en la que no se diferencian los grupos de actores básicos referidos anteriormente. La identificación de dichos grupos solamente servirá a efectos de explicación del criterio por el que se identifica al actor. Se aconseja, por motivo de claridad, que los actores del mismo grupo aparezcan junto en las filas y columnas de la tabla anterior. En el ejemplo, las tres primeras posiciones la ocupan los resultados de la clasificación mediante su perfil en el sistema y las tres últimas los resultados de la clasificación mediante su carácter investigador.

Una "X" en la intersección de una fila y una columna de la tabla indica que los actores correspondientes son incompatibles. El "-" indica la imposibilidad de indicar incompatibilidad entre un actor y él mismo. Así, se observa que en los resultados de la clasificación mediante el perfil, los actores son disjuntos. Mientras que en la clasificación según el carácter investigador no. De esta forma, podría haber un usuario que desempeñara los roles de etnólogo y arqueólogo, por ejemplo.

Pero estas clasificaciones realmente proporcionan los actores básicos del sistema. Sin embargo, en el sistema de los bienes muebles, como en la mayoría de los sistemas de información global aparecen roles derivados de interés. Un actor derivado es todo actor que se puede definir a partir de los actores básicos o de otros actores derivados, como conjunción de los roles correspondientes a los actores componentes. El rol asociado a un actor derivado asumirá los roles correspondientes a los actores que lo componen.

Así, para nuestro ejemplo, un usuario no tiene porqué ser etnólogo o artista. Puede tomar ambos roles a la vez. La diversidad de actores derivados va a tomar un interés especial a la hora de capturar los requisitos funcionales y los de interacción, puesto que por ejemplo, un actor que tome los roles de artista y etnólogo y sea un usuario del centro investigador a la vez, no verá los mismos datos ni podrá realizar las mismas operaciones, que un usuario de internet que sea arqueólogo.

Para realizar la definición de los actores derivados se utilizará una tabla de doble entrada como la tabla 16.

Cada columna corresponderá a un actor, básico o derivado, que forme parte de al menos un actor derivado. De esta forma, encontramos en las primeras columnas a los actores básicos y en las cuatro últimas a actores derivados definidos en la misma tabla. Así el actor ArqArEt representa a un usuario que actúa en el sistema como arqueólogo, como artista y como etnólogo. Cada fila corresponderá, como se ve, a un actor derivado.

El símbolo "^" permite indicar que el rol correspondiente al actor de la columna forma parte del actor correspondiente a la fila. El "-" indica que un actor derivado no puede ser parte componente de él mismo. Así, un actor ArqEtC es un catalogador que actúa en el sistema como arqueólogo, etnólogo y con perfil de catalogador.

Esta complejidad requiere un importante estudio sobre qué información se muestra y cómo se ve la información por cada uno de los actores. Por ello, esta definición de actores será crítica para definir los requisitos funcionales y de interacción ya que, lo que se puede hacer con el sistema y cómo se le van a presentar al usuario estos datos, va a depender directamente del tipo de usuario con el que estemos trabajando.

Actor	Arqueó-logo	Artista	Etnólo-go	Usuario Centro	Catalo-gador	ArArq	ArEt	ArqEt	ArqArEt
ArArq	^	^				-			
ArEt		^	^				-		
ArqEt	^		^					-	
ArqArEt	^	^	^						-
ArArqU				^		^			
ArEtU				^			^		
ArqEtU				^				^	
ArqArEtU				^					^
ArArqC					^	^			
ArEtC					^		^		
ArqEtC					^			^	
ArqArEtC					^				^

TABLA 16: ACTORES DERIVADOS

Otra de las relaciones que se pueden detectar entre los actores es la de generalización/especialización. Un actor especializado es todo actor que se puede definir a partir de los actores básicos o de otros actores especializados mediante una relación de generalización [Jacobson 1999]. El rol asociado a un actor especializado heredará los roles asociados a los actores de los que se especializa y podrá añadir semántica específica al rol que desempeña. La definición de los actores especializados se realizará mediante la notación de generalización de UML y según el patrón mostrado en la tabla 14 añadiéndole el campo *hereda de* en el que se indica el identificador del actor, ACT-X, del que hereda.

En el ejemplo de los bienes muebles no aparece ninguno de estos casos, pero no es poco común encontrarlo en los sistemas de información global.

Actividad 4: Identificar y definir los requisitos funcionales

Los requisitos funcionales van a responder a la pregunta de *¿qué podrá hacer el sistema con la información que almacena?*. Esta pregunta nos la responden los casos de uso, técnica que se usará para capturar estas necesidades. Estos diagramas se describen mediante una información gráfica (diagramas de casos de uso) y una información textual. En los casos de uso aparecen dos elementos importantes, el caso de uso en sí y

los actores. Los actores se definieron en la actividad anterior, así que aquí se hará referencia a esas definiciones.

Para recoger la definición de los casos de uso parece adecuado el patrón propuesto en [Durán 2000] que se asumirá en esta propuesta. Sin embargo, en nuestra propuesta se hace la especificación de los casos de uso en base a los actores que se definieron en la etapa anterior. Es decir, al definir un caso de uso se indicará una lista de todos los actores que pueden ejecutar ese caso de uso. De esta forma se obtiene una visión funcional del sistema orientada a los actores.

Para nuestro ejemplo de los bienes muebles vemos en la figura 33 uno de los casos de uso. Es el caso de uso que representa que un usuario da de alta un nuevo bien mueble.

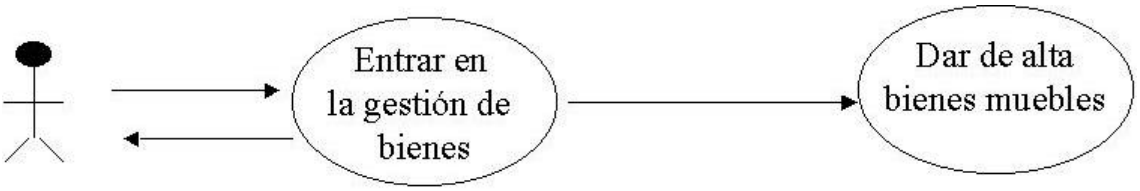

FIGURA 33: EJEMPLO DE CASO DE USO

Para que la definición de este caso de uso sea completa, es necesario acompañarla de un patrón que lo defina de forma más concreta. El patrón asociado a este caso de uso se presenta en la tabla 17.

RF-01	Dar de alta bienes muebles	
Objetivos asociados	?? OBJ-01: Gestionar los bienes muebles	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el usuario pulse la opción de dar de alta a los bienes	
Precondición	El usuario debe estar dado de alta en el sistema	
Actores	?? AC-04: Usuario Centro ?? AC-05: Catalogador	
Secuencia normal	**Paso**	**Acción**
	1	Ejecutar el caso de uso de Entrar en la gestión de Bienes Muebles
	2	El sistema da entrada al usuario
	3	El usuario indica la provincia, el municipio y el inmueble en el que se ubica el bien
	4	El sistema solicita la denominación
	5	El usuario introduce la denominación
	6	El sistema da de alta el bien asignando un nuevo código y permite al usuario editar todos los datos referentes al bien
	7	El usuario introduce los datos que desee y solicita salir
	8	El sistema vuelve al menú inicial
Postcondición	ninguna	
Excepciones	**Paso**	**Acción**
	1	El usuario no está recogido en el sistema, por lo que se apaga el acceso al mismo
	5	El bien ya estaba dado de alta. Se emite un mensaje al usuario y se abandona la aplicación
Frecuencia esperada	10 veces /mes	

TABLA 17: EJEMPLO DE PATRÓN PARA UN CASO DE USO

En este ejemplo se puede ver cómo hay que asociar el caso de uso a una lista de objetivos y cómo hay que indicar la secuencia normal de trabajo que éste tiene. Además habrá que indicar qué roles pueden asumir el papel del actor que participa en él y las

precondiciones y postcondiciones que deben cumplirse antes y después de su ejecución. Así como las excepciones que se pueden producir durante su ejecución.

Nuevamente también es este patrón más completo en la propuesta genérica. Como vemos aquí se indica nuevamente qué objetivos se cubren al realizar este caso de uso. Además se especifica de forma clara la secuencia de pasos a seguir para que se pueda realizar la operación de dar de alta un bien.

Actividad 5: Identificar y definir los requisitos de interacción

Llegados a este punto, ya se ha recogido qué hay que almacenar en el sistema, quién va a usar el sistema y qué se puede hacer en el sistema. Sin embargo, para los sistemas de información global, la interfaz es un aspecto crítico que es fundamental en el desarrollo.

En el entorno actual, una aplicación no sólo debe ofrecer al usuario toda la información y funcionalidad que éste le pida, debe además darle esa información y funcionalidad en el momento apropiado y de la forma apropiada. El orden el que se muestre la información o la forma en que se muestre es un aspecto crucial.

En esta actividad se van a definir lo que conoceremos como requisitos de interacción. Un requisito de interacción va a ser una forma de representar como se va a mostrar al usuario la información. Basándose en criterios establecidos por el cliente, los datos concretos de los requisitos de almacenamiento de información se van a mostrar agrupados en diferentes prototipos de visualización.

Además, cada uno de estos requisitos de interacción llevará asociado una funcionalidad. La funcionalidad vendrá dada por cada uno de los requisitos funcionales que se puedan ejecutar en ese requisito de interacción.

Los requisitos de interacción van a estar compuestos por dos aspectos: los criterios de búsqueda y los prototipos de visualización. Los primeros van a usar un lenguaje seminatural en el que se va a recoger cómo el usuario quiere recuperar la información. Para ello, se usará una propuesta de lenguaje denominado BNL (Bounded Natural Language) [Brisaboa 1999], que mediante la definición de frases indicará como el usuario va a cuestionar al sistema.

Los segundos van a definir cómo se mostrará la información al usuario y la funcionalidad que tiene asociada esa muestra de información. Para definir estos requisitos en el documento de requisitos, se hará uso de dos patrones: uno para las preguntas y otro para los prototipos de visualización. Los primeros recogerán las frases escritas en BNL, presentarlos obligaría a explicar esta técnica, que queda un poco fuera del objetivo del documento.

Un ejemplo de patrón para los prototipos de visualización se recoge en la tabla 18. Este es el prototipo de visualización describe como se va a interactuar con el usuario para mostrarle los datos de un bien mueble. Se va a recoger qué datos se les mostrará y como puede entrar y salir de la pantalla de alta de un bien. Además se indica qué actores podrán ver la pantalla y qué datos verá cada uno.

El campo actor/es es quizás uno de los más relevantes. En él se recoge la lista de los identificadores de actores que podrán hacer uso del requisito. Estos actores pueden ser derivados o básicos, pero han debido de ser definidos en la actividad 3. Aquí como vemos pueden hacer uso del requisito los arqueólogos, los etnólogos, los artistas, los

catalogadores, los usuarios de internet o los usuarios del centro. Cualquier actor que tenga al menos alguno de estos roles podrá interactuar en este requisito.

PV-01	Pantalla de datos de un bien mueble
Objetivos asociados	OBJ-01: Gestión de los bienes muebles
Actor/es	AC-01: Arqueólogo AC-02: Etnólogo AC-03: Artista AC-04: Catalogador AC-05: Usuario del Centro AC-06: Usuario de internet
Descripción	El sistema deberá permitir la visualización de los datos concretos que se muestran a continuación y la navegación expresada.
Condición	El código debe estar dado de alta o ser nulo
Frases	?? PR-01
Funcionalidad asociada	Si Actor = Catalogador o Actor = Usuario del Centro ✍ RF-01: Dar de alta bienes
Información visualizada	✍ RA-01.Código del bien ✍ RA-01.Denominación ✍ RA-01.Otra denominación ✍ RA-01.Titulado/Representado ✍ RA-01.Ubicación habitual ✍ RA-01.Ubicación temporal ✍ RA-01.Acceso ✍ RA-01.Cronología ✍ RA-01.Certeza ✍ RA-01.Imágenes ✍ RA-01.Períodos Históricos ✍ RA-01.Estilo ✍ RA-01.Tipologías Si Autor = Artista ✍ RA-01.Autor
Salidas	PV-12
Entradas	PV-12 PV-11

TABLA 18: PATRÓN PARA LA RECOLECCIÓN DE PROTOTIPO DE VISUALIZACIÓN

Otro campo interesante es el de frases. Aquí hay una variación en base a si el prototipo es para un único actor o para varios. Si es para uno solo, se indican las frases teniendo en cuenta que las frases debieron ser definidas para el actor en cuestión, pero si hay varios actores puede ser que las frases difieran de uno a otro. Es este caso, en este campo se colocará una sentencia condicional que indique como sería el campo para cada actor.Por ejemplo podríamos pensar en un caso de prototipo para los actores AC-01, AC-02, AC-03 y AC-04 cuyas frases difieren para los dos primeros y son iguales para los dos últimos, el patrón quedaría como se muestra en la tabla 19.

El campo de las condiciones permite indicar las condiciones que los datos mostrados en el requisito de interacción van a cumplir. Se definen basándose en los valores de los parámetros de entrada. La definición de estos datos puede hacerse en lenguaje natural o en un lenguaje más formal que sea definido por la organización. Igual que en el campo anterior, este campo se puede enriquecer con expresiones condicionales que permitan individualizar el campo para cada actor, de la misma forma que el caso de los parámetros de entrada.

En la funcionalidad asociada se recoge cada uno de los requisitos funcionales que se van a poder ejecutar desde el requisito de interacción. Este campo, como en los

anteriores, puede estar condicionado a los parámetros de entrada que haya o a los actores definidos. En nuestro ejemplo vemos que si tenemos un catalogador o un usuario del centro podrán ejecutar el requisito de dar de alta un nuevo bien.

El campo más relevante del patrón es el de información visualizada. En él se mostrarán, en orden de aparición, cada uno de los datos que el requisito va a mostrar. Para ello, se hace referencia al requisito de almacenamiento que lo guarda y al dato concreto de éste que se muestra. Igualmente, estas referencias a los datos concretos se podrán enriquecer con estructuras condicionales que permitan recoger las pequeñas diferencias entre los actores.

Los campos de salida y de entrada recogen desde qué otros requisitos de interacción se puede llegar al que se define en el patrón y a qué otros se puede ir desde éste respectivamente. Con esto se tiene definida la navegación por el sistema. En nuestro ejemplo desde este requisito podríamos navegar al catalogado como RI-12 y a él podríamos acceder desde el RI-11 y el RI-12, que deberán ser definidos en el documento de requisitos.

Actores	ACT-01 ACT-02 ACT-03 ACT-04
Descripción	El sistema deberá permitir la visualización de los datos concretos que se muestran a continuación y la navegación expresada.
Parámetros de entrada	SI (actor = AC-01) <Frases para el actor AC-01> ACT-01> SI (actor = AC-02) <Frases para el actor AC-02> EN OTRO CASO <Frases para el resto de actores>

TABLA 19: EJEMPLO DE ESTRUCTURA CONDICIONAL

Actividad 6: Identificar y definir los requisitos no funcionales

Para completar la especificación de requisitos, es necesario revisar de nuevo los objetivos para determinar otros requisitos que por su carácter no hayan sido catalogados anteriormente. Algunos de estos requisitos serán los requisitos de comunicaciones, de fiabilidad, etc. Para recoger estos requisitos se hará uso de lo indicado en la *Metodología para la Elicitación de Requisitos*[Durán 2000]. En ésta, los requisitos aparecen nuevamente definidos mediante un patrón que recoge toda la información necesaria para describirlos.En la tabla 20 vemos un ejemplo para los bienes muebles en el que se recoge la necesidad de que el sistema permita hacer backup.

RNF-01	Copia de seguridad
Objetivos asociados	OBJ-05: Conseguir un sistema estable y seguro para la catalogación de los bienes
Descripción	El sistema deberá ofrecer la posibilidad de realizar backups periódicos de la información, así como la recuperación en caso de pérdidas de la información.
Comentarios	Este requisito es esencial porque hay información reservada de los bienes

TABLA 20: EJEMPLO DE REQUISITO NO FUNCIONAL

6.3.3. Análisis

El flujo de trabajo de análisis de sistema tiene esencialmente tres diferencias con respecto a la mayoría de las propuestas. Éstas son, como ya hemos enunciado: los diferentes medios de almacenamiento de información, la importancia de la interfaz y los aspectos navegacionales. Por ello, habrá que incluir el estudio de estos aspectos en el análisis para después pasarlos e incluirlos en el diseño.

De esta forma, en análisis debemos conseguir un modelo de clases que represente al sistema. Este modelo irá acompañado por un modelo dinámico cuando resulte necesario, así como por una estructuración en paquetes cuando su complejidad sea alta. Además, en análisis se hará un refinamiento de los casos de uso para concretizarlos y asignar responsabilidades y participaciones de las clases de análisis. También se consolidará la navegación y se propondrán los primeros prototipos de interfaz.

De forma esquematizada, el flujo de trabajo de análisis debería asumir las siguientes actividades:

- Actividad 1: Construir un modelo conceptual de análisis.
  - Tarea 1.1- Identificar y definir las clases del sistema.
  - Tarea 1.2- Agrupar las clases del modelo en paquetes y establecer relaciones entre ellos si es necesario.
  - Tarea 1.3- Realizar el refinamiento de los casos de uso para concretizarlos y asignar responsabilidades y participación en ellos de las clases de análisis.
  - Tarea 1.4- Definir el modelo dinámico de las clases de análisis
- Actividad 2: Refinar el modelo de navegación del sistema.
- Actividad 3: Definir los prototipos de interfaz.

A continuación vamos a ver cada una de estas actividades. Sin embargo, hay que tener en cuenta que este flujo de trabajo esta actualmente en fase de desarrollo. Por ello, la aplicación al ejemplo y la definición de estas actividades no va a estar tan detallada como las del flujo anterior.

Actividad 1: Construir el modelo conceptual del sistema

Partiendo de los requisitos de almacenamiento de información y de la definición de los actores, se debe realizar el modelo conceptual del sistema. Esto consiste en realizar un modelo de clases, que mediante la nomenclatura de UML represente la estructura conceptual de la aplicación.

Los patrones de los requisitos de almacenamiento de información y de definición de naturaleza de los medios, definidos en la actividad 1 del flujo de definición de requisitos, es una definición estructura que está pensada para que, a partir de ella se pueda obtener de una forma mecánica, el modelo conceptual del sistema.

No vamos a entrar aquí a definir cada una de las pautas a seguir para conseguir este modelo a partir de nuestros patrones definidos, pero podemos indicar que a groso modo, cada requisito de almacenamiento y cada naturaleza va a traducirse a una clase, cuyos atributos van a ser los datos concretos. Además a partir de estos datos, cuando su naturaleza sea otro requisito, una nueva naturaleza o cuando su cardinalidad sea mayor a 1, se van a definir las asociaciones entre estas clases.

Una vez conseguido, como se ha comentado, el modelo de clases básico, se debe estudiar éste para detectar si es necesaria la división en paquetes del mismo.

Además, será necesario estudiar los casos de uso que se vieron en el flujo de especificación de requisitos, para definir los métodos asociados a cada clase. También existirán en la metodología que estamos desarrollando pautas para conseguir esta definición.

Realizando estas tareas y tomando únicamente la definición del requisito de almacenamiento RA-01 que vimos en el apartado anterior, podríamos obtener un modelo de clases similar al que se muestra a continuación en la figura 34. Estudiando cada requisito del sistema iríamos consiguiendo ampliar este modelo.

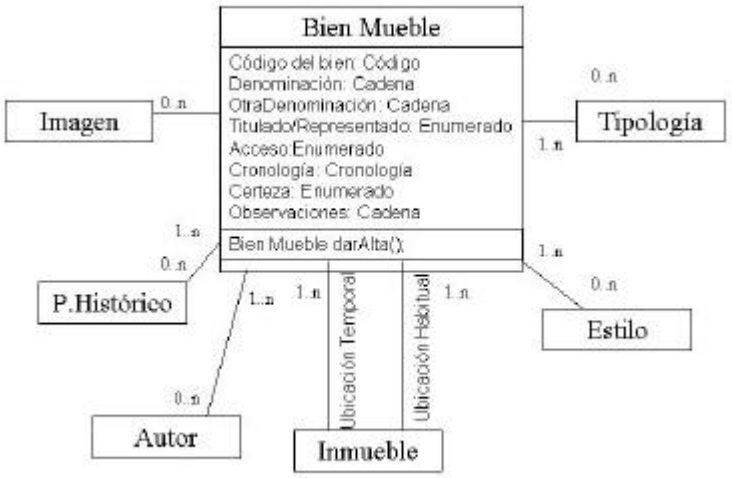

FIGURA 34: DIAGRAMA DE CLASES DE ANÁLISIS

También a partir de los casos de uso, se puede detectar aquellas clases que tienen asociado un determinado dinamismo.

El dinamismo de cada clase que tenga asociado un modelo dinámico vendrá recogido en el documento de análisis a través de un diagrama de estados [Jacobson 1995]. En la figura 35 vemos el que correspondería a la clase bien mueble de nuestro ejemplo

Al igual que en el flujo de trabajo de especificación de requisitos, el flujo de análisis lleva asociado un documento en el que se recogen los resultados de éste. En dicho documento se recogen no solo los diagramas de clases, de paquetes y de diagramas de estados, sino que además se debe recoger un diccionario de datos que elimine la ambigüedad que puede aparecer en estos modelos. En la metodología se presentará también el formato de este diccionario. No vamos a entrar en definirlo por la generalidad de este documento.

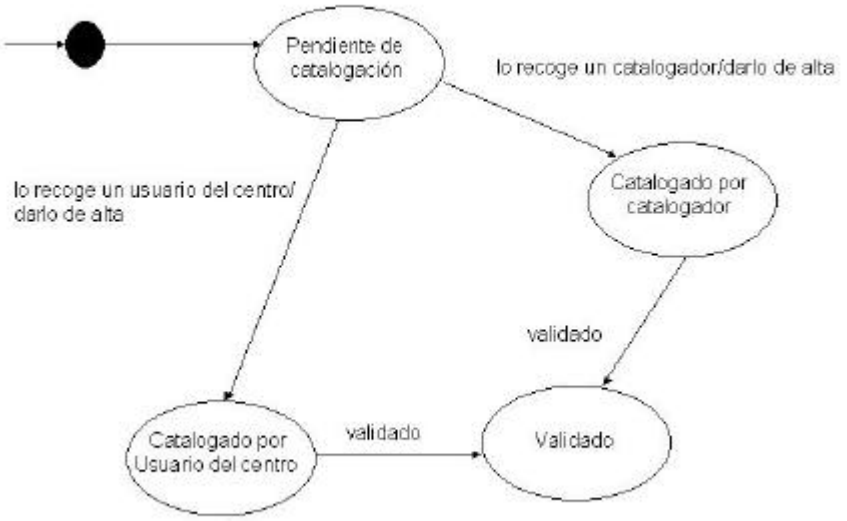

FIGURA 35: DIAGRAMA DE ESTADOS PARA LOS BIENES MUEBLES

Actividad 2: Refinar el modelo de navegación del sistema

Cuando ya se ha representado el modelo conceptual del sistema, es necesario estudiar y refinar el modelo navegacional del sistema que estamos definiendo.

En los requisitos de interacción se capturó cómo se va a mostrar la información al usuario y cómo se puede navegar a través de estos módulos de información. Sin embargo, para que la estructura de la navegación sea consistente debe cubrir una serie de aspectos. Por ejemplo, el usuario debe tener la posibilidad de llegar a toda esta información desde un punto que podríamos definir como pantalla inicial. Se debe garantizar que la navegación es consistente. Para ello, en esta actividad se definen lo que se denominarán nodos y clases de navegación. Los nodos serán elementos que contienen información sobre un determinado aspecto. Prácticamente se van a corresponder con cada requisito de interacción. Por su parte, las clases de navegación van a ser elementos que van a conectar los diferentes nodos que se definan. Van a ser por ejemplo menús, diccionarios o rutas que indiquen cómo va a conectarse la información, definiendo así la estructura de navegación del sistema.

Para definir estos elementos nuevamente se va a ser uso de los patrones de manera que quedarán definidos de forma estructurada y completa.

Además, la metodología ofrecerá una serie de pasos para conseguir estos elementos.

Una vez definidos estos elementos, habrá que estudiar la consistencia del modelo de navegación. Para eso también se propondrán técnicas que de forma metódica permiten asegurar esta consistencia, indicando si el modelo es completo y adecuado a las necesidades.

Actividad 3: Generar los prototipos de interfaz

Partiendo de los requisitos de interacción y de los elementos de navegación definidos en la actividad anterior se va a hacer una definición de la interfaz de la aplicación.

Estos prototipos de interfaz no se van a preocupar de los aspectos estéticos de la aplicación, no se van a preocupar del colorido o de si va un botón o un check box en tal lugar. Estos prototipos van a mostrar qué información se ofrece al usuario y en qué orden se ofrece, así como las navegaciones que se ofrecen.

Por ejemplo, para mostrar algunos de los datos de los bienes podríamos definir el prototipo que se muestra en la figura 36.

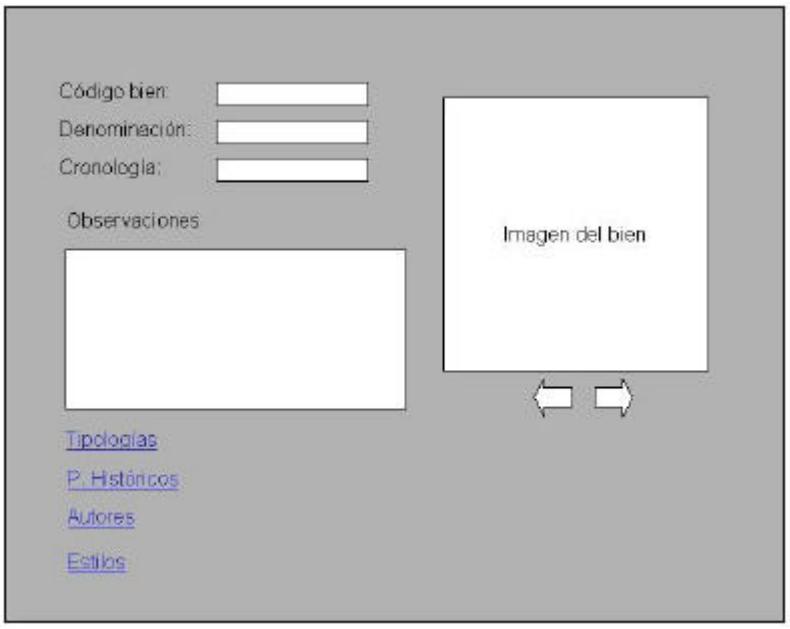

FIGURA 36: EJEMPLO DE PROTOTIPO DE INTERFAZ

Nótese que aquí no aparecen aspectos de diseño. Se indica que al usuario se le va a mostrar el código, la denominación, la cronología y las observaciones de un bien en un determinado orden.

Indica además que se muestran las imágenes de los bienes. Nótense las dos flechas debajo del cuadro de imagen. Esto indica que puede haber más de una asociada a un bien.

También se recoge en esta pantalla que desde aquí podemos acceder a las tipologías, los períodos históricos, los autores y los estilos del bien. O que podemos acceder al menú inicial. Pero no se recoge ningún aspecto visual.

6.3.4. Diseño

En el flujo de trabajo de diseño se parte del modelo de clases de análisis conseguido en el flujo anterior, así como del análisis hecho a los casos de uso y de la agrupación en paquetes de análisis. A partir de estos productos se realiza el diseño de la arquitectura del sistema, hacer un diseño de los casos de uso, conseguir un modelo de clases de diseño y realizar una división del sistema en subsistemas. Pero además, hay que recoger y diseñar los aspectos de navegación y de interfaz de usuario.

✍ Actividad 1: Diseñar el modelo básico.

- Tarea 1.1- Diseñar la arquitectura del sistema.
- Tarea 1.2- Diseñar los casos de uso y los requisitos no funcionales.

- Tarea 1.3- Dividir el sistema en subsistemas.
- Tarea 1.4- Diseñar el modelo de clases básico de análisis.

✍ Actividad 2: Diseñar el modelo navegacional.

- Tarea 2.1- Diseñar el modelo de clases navegacionales.
- Tarea 2.2- Diseñar los contextos navegacionales.

✍ Actividad 3: Diseñar la interfaz abstracta.

- Tarea 3.1- Diseñar los prototipos de pantallas.
- Tarea 3.2- Diseñar el sistema dinámico de pantallas.

Como se ha venido comentando, la fase de diseño no está tan elaborada como las anteriores. Simplemente vamos a marcar las ideas de lo que hay que conseguir en cada actividad.

Actividad 1: Diseñar el modelo básico

El diseño del modelo básico consistirá en obtener el diseño del sistema pero dejando al margen todo lo que serían los aspectos de navegación y de interfaz. La idea de separar estos aspectos del diseño, es asumida por la mayoría de los autores que hemos estudiado, esto facilita la reutilización y el mantenimiento. Para un mismo diseño básico puede haber muchos diseños de navegación y cada uno de estos puede tener definidas diferentes interfaces. De esta forma si cambiamos la interfaz o la navegación el modelo básico puede quedar igual.

En el diseño básico, una de las tareas a realizar es el diseño de la arquitectura. Para realizar el diseño de la arquitectura en un sistema de información global, hay que tener en cuenta que los soportes de estos sistemas, en la mayoría de los casos, se caracterizan por estar dispersos en la red, por estar implantados en sistemas heterogéneos y por la ausencia de estándares para el acceso a los mismos. Por ello, las propuestas de arquitecturas que se den en el marco de estos sistemas, deben contemplar los siguientes aspectos:

✍✍Almacenamiento

✍✍Clasificación

✍✍Interfaces para la presentación de resultados

✍✍Distribución del contenido de la biblioteca al usuario final

✍✍Administración y control de acceso.

En principio, el diseño de la arquitectura no debe verse afectado por aspectos de navegación o interfaz, que serán objetivos de los subflujos siguientes.

La metodología no puede ofrecer una opción genérica y válida para todos los sistemas existentes, sin embargo tiene como objetivo el ofrecer algunas de las actuales [Brisaboa 1999] [Cordero 2000] e indicar cómo a partir de los resultados anteriores se pueden obtener éstas de una forma casi automática.

Otra tarea importante del diseño básico es la de analizar la funcionalidad de las clases que se han obtenido en el modelo de clases del análisis. Por ello, en el diseño básico también se afrontará el diseño de los casos de uso. Durante la especificación de requisitos y en el análisis se han establecido los casos de uso que representan los requisitos funcionales de la aplicación. A la hora de diseñar estos casos de uso es

conveniente tener en cuenta los métodos de diseño típicos de la Ingeniería del Software: patrones de diseño, técnicas para recoger los aspectos de concurrencia y persistencia, etc.

En el diseño de estos casos de uso, hay que hacer un hincapié especial en algunos aspectos muy importantes para el entorno de los sistemas de información global. Así cuando diseñamos estos casos de uso debemos identificar la concurrencia inherente en el sistema. Esta tarea en el entorno de los sistemas de información global, y en especial en aquellas que se distribuyan por internet, es esencial, puesto que deben controlar la concurrencia. En nuestro sistema ejemplo, podemos tener a la vez personas realizando modificaciones y consultas al mismo tiempo, sin que tengamos la posibilidad de saber el número de usuarios, puesto que al distribuirse por Internet, esto es imposible.

A la hora de estudiar la concurrencia, hay también que identificar los recursos globales y determinar los mecanismos para controlar el acceso a los mismo. Partiendo del hecho de que los sistemas de información global en muchos casos suelen distribuirse por internet y que, en principio, a ellas puede acceder cualquiera, es necesario recortar y/o controlar a los usuarios que pueden usarlas.

Para nuestro ejemplo, y partiendo de los tres tipos de usuarios que definimos en la especificación de requisitos en base al perfil funcional, una posible propuesta de solución sería que los usuarios que pueden dar de alta o modificar cualquier dato: los usuarios catalogadores y del centro, accedieran al sistema a través de claves y los usuarios que accedan al sistema sin clave sólo podrán usar la función de consulta, caso que se correspondería con los usuarios de internet.

Cuando diseñamos los casos de uso y conocemos las necesidades de almacenamiento que hay que tener, es hora de seleccionar la estrategia básica para implementar los almacenes de datos en términos de estructuras de datos, archivos y bases de datos. Esta tarea adquiere una importante relevancia en los sistemas de información global. Ya hemos hablado de los diferentes tipos de datos que se manejan en estos sistemas: imágenes, sonidos, documentos, animaciones, etc. Es necesario pues, buscar sistemas gestores de bases de datos y soportes que nos permitan un almacenamiento y un tratamiento eficiente y seguro de todos los tipos de datos con los que vamos a trabajar.

Debido a la multiplicidad de medios y naturalezas que se usan, no siempre es posible encontrar un sistema capaz de dar soporte a todos ellos. Las bases de datos de ultima generación que permiten almacenar y recuperar imágenes o sonido de la misma forma que se recuperan los tipos básicos como los texto o booleanos, están sólo naciendo. Por ello, aquí podemos proponer múltiples sistemas de almacenamiento que den soporte a nuestras necesidades. Por ejemplo, para la biblioteca de bienes muebles, vamos a proponer para todos los datos el uso de una base de datos relacional clásica y las imágenes las vamos a mantener almacenadas en una zona de memoria del servidor, de manera que en la base de datos tengamos una referencia a la ruta donde podemos encontrar las imágenes asociadas a un bien. Este diseño, que podría parecer un poco arcaico es, sin duda, el que más se asume cuando hay que diseñar bases de datos con datos gráficos, debido a que es el método más barato y sencillo de diseñar.

En relación a la multiplicidad de medios, no solo hay que estudiar cómo se van a almacenar, hay que estudiar también las necesidades de consulta de la información. Esta tarea está muy relacionada con la anterior en el sentido de que los sistemas de información global cada vez requieren sistemas de consulta más elaborados y complejos. La complejidad de estos sistemas de búsqueda no viene dada exclusivamente por el hecho de que los volúmenes de la información sean muy grandes.

La complejidad también aparece en la idea de que las búsquedas sobre información no textual cada vez son más necesarias. Así, en el entorno de nuestra aplicación podríamos pensar en buscar todas las imágenes de los bienes que tengan un crucifijo en alguno de sus adornos. Pero en el entorno de los bienes muebles, así como en el de los sistemas de información global en general, aparecen otras consultas también complejas que pudieran permitir al usuario hacer preguntas imprecisas. Por ejemplo, el hecho de detectar los bienes muebles desarrollados en un entorno del siglo XVIII y que se encuentren en yacimientos próximos a Sevilla.

Por último, también en el diseño básico, habrá que estudiar las condiciones de contorno. En las aplicaciones con las que trabajamos es esencial el contemplar las condiciones de contorno: qué ocurre al iniciarse la aplicación y en su caso cómo se realiza la conexión inicial al servidor; qué ocurre cuando finaliza la aplicación y la desconexión del servidor; y sobre todo que ocurre cuando se producen errores. En las aplicaciones que estamos tratando es necesario contemplar esos errores que se producen cuando hay caídas de sistemas de comunicaciones. Pero cuando estamos tratando con aplicaciones que se ejecutan en equipos diferentes a través de internet, por ejemplo, es necesario hacer un estudio exhaustivo de otro tipo de errores, que, hoy por hoy aún son un problema. Al distribuirse la aplicación por internet, no podemos controlar aspectos como el navegador que va a usar el usuario y las utilidades que éste ofrece; las capacidades del equipo que va a conectarse o la velocidad de la conexión. Por esta razón, es necesario controlar estos tipos de errores en la fase de diseño, sobretodo cuando trabajamos con animaciones o datos en los que las características temporales y de sincronización tengan una vital importancia. Así, para nuestra aplicación habría que controlar, por ejemplo, que las imágenes asociadas a los bienes se visualicen correctamente o en su caso, que si el sistema cliente no puede visualizar las imágenes, que se muestre un mensaje indicando que en ese lugar habría una imagen.

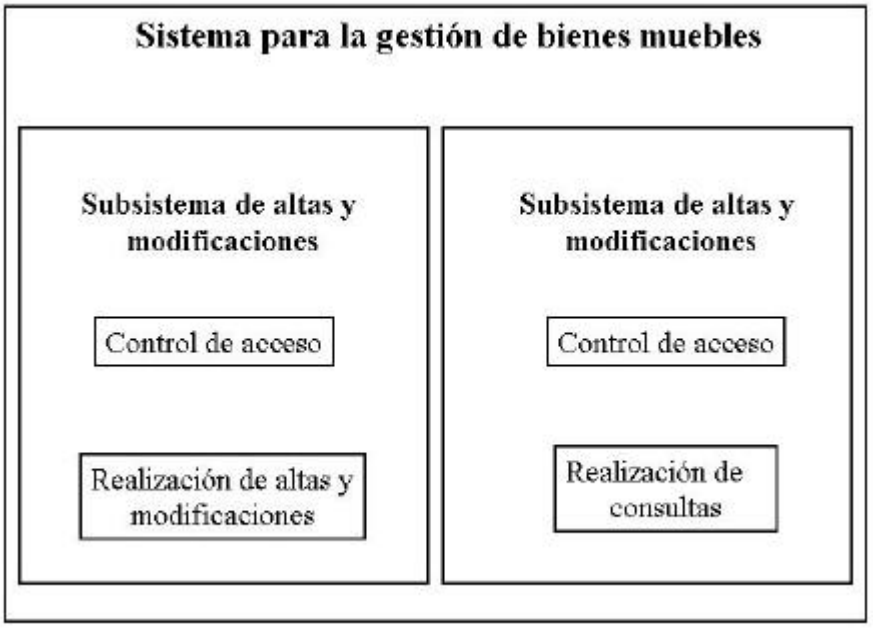

FIGURA 37: DIVISIÓN EN SUBSISTEMAS PARA EL EJEMPLO

También es en la actividad de diseño básico donde hay que plantearse el organizar el sistema en subsistemas. Debido a la complejidad funcional que suelen tener los sistemas

de información global, es bastante aconsejable el dividir el problema inicial en subproblemas más sencillos, usando así una técnica de divide y vencerás. Para nuestro sistema vamos a plantearnos dos subsistemas principales: el que permite realizar consultas y el que permite realizar altas y modificaciones. Cada uno de ellos tendrá a su vez dos tareas fundamentales. Una que controle los permisos del usuario y otro que controle lo que son las tareas de alta, consulta y modificación propiamente dichas. En modo de diagrama quedaría como se muestra en la figura 37.

La metodología propondrá un diccionario de datos para estos módulos, de manera que se describa cada uno de ellos de forma estructurada. Además describirá el lenguaje gráfico para representar la división modular.

Una vez definidos los subsistemas, habrá que asignar cada uno de ellos a procesadores y a tareas. Otra de las característica de los sistemas de información global es que suelen ser aplicaciones distribuidas, por lo que será necesario establecer dónde se va a realizar cada tarea. En nuestro ejemplo, podríamos pensar en un almacén de datos, localizado en el servidor central y una aplicación que se ejecuta en equipos remotos y que se encargan de controlar las peticiones de los usuarios. Una vez filtradas y comprobadas las consultas y modificaciones, los equipos remotos pedirán al servidor central que realice la consulta, la modificación o el alta.

Por último, en el diseño básico hay que plantearse el conseguir el modelo de objetos básico de diseño. De la misma forma que hemos conseguido un modelo que represente al sistema en la fase de análisis, vamos a partir de ese modelo de análisis y vamos a enriquecerlo para obtener el modelo de objetos de diseño. En este modelo, no será necesario aún introducir aspectos de navegación o de interfaz de usuario, puesto que esos quedarán para los subflujos posteriores.

Para conseguir este modelo, será necesario estudiar ciertos aspectos del sistema para:

- Obtener las operaciones para el modelo de objetos a partir de los demás modelos. Esta tarea va ser el primer paso para enriquecer el modelo de objetos obtenido en análisis puesto que va a bajarse a mucho más nivel del que se hacía en análisis. Estas operaciones se van a obtener estudiando el modelo dinámico del análisis y los casos de uso.
- Empaquetar las clases y las asociaciones en paquetes. De la misma forma que en análisis agrupábamos las clases en paquetes que nos permitieran estructurar el sistema, en diseño es aconsejable agrupar las clases en paquetes y establecer relaciones entre ellos. Esta tarea va a facilitar la compresión del modelo y va a hacer más sencillo el trabajo del implementador.
- Retocar el modelo añadiendo los objetos que se estimen necesario para representar aquellos atributos del modelo de análisis que se van a tratar como clases.

La metodología presentará las pautas necesarias para establecer una forma metódica de llegar a este modelo. Pero es un punto aún abierto en este trabajo de investigación.

Para nuestro ejemplo, el modelo no quedaría modificado al aplicar esta fase, quedando como se mostró en la figura 34. Esto es lógico de pensar puesto que no se ha analizado la aplicación completa y que no se han aplicado patrones de diseño.

Actividad 2: Diseñar el modelo navegacional

A la hora de realizar un modelo navegacional, hay que tener en cuenta:

- Qué objetos del modelo básico van a ser navegables. Esto va a estar íntimamente relacionado con los nodos definidos en la especificación de requisitos navegacionales.
- Qué tipo de relaciones y estructuras de composición hay entre estos objetos navegables.
- De los objetos navegables, habrá que estudiar en qué forma se mostrarán dependiendo del contexto en el que nos movamos.
- Las conexiones entre los distintos objetos y las posibilidades de navegación de un objeto a otro. Esto va a estar bastante relacionado con los enlaces entre nodos que se detallan en la especificación de los requisitos navegacionales.

Proponemos dos modelos para recoger los aspectos de navegación: el esquema de Clases Navegacionales y el diagrama de Contexto Navegacional.

El representar el aspecto de navegación mediante un modelo de clases es una idea que muchos autores comparten y que da muy buenos resultados. El esquema de clases navegacionales se obtiene a partir del modelo de clases de diseño obtenido en el diseño básico. El modelo básico se va a modificar de manera que se van a obtener nuevas clases que pueden ser *nodos*, *enlaces* o *clases índice*. De esta forma, una clase del modelo básico puede originar una o varias clases del modelo navegacional. El modelo de clases navegacionales va a estar compuesto por este nuevo modelo de clases que contiene índices, nodos y enlaces.

En principio, una clase nodo es una clase contenedora de información. Va a estar muy relacionada con el concepto de prototipo de visualización. Es una clase que agrupa información de una o más clases conceptuales y que representa cómo se va a agrupar la información para mostrarla al usuario.

Los enlaces serán clases que van a instanciarse en asociaciones entre clases nodos y representarán la posibilidad de navegar desde una clase nodo a otra.

Por último, una clase índice va a ser un menú, un diccionario, etc. que sirva de acceso a las clases nodos.

Estas definiciones tan genéricas deberán ser concretadas en la propuesta completa de la metodología. La versión final de este flujo de trabajo debe indicar cómo conseguir este modelo y cómo definir estas clases.

El otro modelo a realizar, aunque asume el nombre de OOHDM y tiene el mismo objetivo, está previsto que sea algo diferente. El diagrama de contexto navegacional va a indicar cómo varía la navegación en función del contexto en el que se esté navegando. Representar esto es esencial para definir el sistema de navegación de un proyecto software. Sin embargo, y a pesar de que los primeros que proponen la nomenclatura de Contexto Navegacional es OOHDM, su propuesta resulta muy compleja cuando los sistemas son muy grandes. Uno de los objetivos de la propuesta es buscar nuevos modelos o nuevas versiones de modelos que permitan representar los contextos navegacionales de una forma más intuitiva.

Actividad 3: Diseñar la interfaz abstracta

Una vez que se tienen diseñado los aspectos de navegación, pasamos a la especificación de los aspectos de interfaz. El hecho de separar los aspectos de navegación de los aspectos de interfaz permite diseñar distintos interfaces para un mismo modelo navegacional.

El diseño de la interfaz abstracta se refiere a revisar los prototipos que se han definido en la fase de análisis y enriquecerlos con todos los nuevos aspectos de diseño conceptual y navegacional. Para el diseño de la interfaz abstracta se propone el uso de las vistas abstractas de datos [Cowan 1995] (ADVs). Las ADVs son objetos que tienen un estado y un interfaz. Son objetos abstractos que sólo representan el estado y la interfaz, sin hacer referencia a la implementación. Mediante el uso de los ADVs representaremos la estructura de la interfaz, cada ADV representa la interfaz que se muestra al usuario de un objeto de navegación, de una parte de un objeto del modelo navegacional o de un conjunto de ellos. Sin embargo, entre los ADVs y las clases del modelo navegacional que tengamos en nuestro sistema del subflujo anterior, se establecen relaciones que hay que recoger. Para expresar estas relaciones se usará el diagrama de configuración y los diccionarios de datos de los ADVs.

Si a todo esto añadimos que debemos recoger el dinamismo que hace cambiar las interfaces, es decir, como cada ADV responde a los eventos externos, aparece la necesidad de usar una nueva técnica que nos permita recoger el dinamismo del sistema de interfaces. Esta técnica la constituyen los diagramas de estado.

Esta es la idea general que se propone para esta actividad. Sin embargo hay algunos aspectos que hay que revisar en ella.

Por un lado, un punto abierto al estudio de los ADVs es la necesidad de normalizar la nomenclatura usada para representarlos. La propuesta metodológica, en su versión final, debe incluir una normalización de esta nomenclatura así como las pautas a seguir para conseguir los ADVs en función de los prototipos de análisis y los modelos de diseño conseguidos en el diseño conceptual y navegacional. En principio el resultado sería algo similar a los prototipos de interfaz del análisis, pero aquí ya recogeríamos aspectos funcionales y la estructura de la navegación estaría más retocada con los aspectos del diseño navegacional.

Los ADVs, además irán acompañada de un diccionario de datos y un diagrama que se denomina diagrama de configuración, en los que se indicara qué actor o actores pueden hacer uso de ella y la relación que existe entre los atributos, enlaces y aspectos funcionales que aparecen en ella y entre el modelo navegacional. La nomenclatura de este diagrama y de los diccionarios de datos es otro aspecto que debe ofrecer la metodología y que aún no está desarrollado.

El diseño de las ADVs representa la estructura dinámica de la intefaz. Sin embargo existe la necesidad de recoger como el usuario interactúa con el sistema. Por ello, es necesario enriquecer este diseño con las máquinas de estado de las ADVs y de las clases navegacionales que se vean afectados por la interacción con el usuario. Esto nos serviría para representar cómo cada uno de estos ADVs reaccionan ante los eventos y cómo afecta esto al sistema navegacional establecido. Aunque los diagramas de estados es ya una técnica consolidada para la representación del dinamismo, es necesario que la metodología estudie la adecuación al diseño de interfaz abstracta en los sistemas de información global. Esta es otra tarea de las que queda pendiente.

6.3.5. Implementación

Llegar a un programa ejecutable tras la realización de las fases anteriores no es algo complejo. En este flujo de trabajo, la tarea fundamental es conseguir un programa ejecutable que implemente a los modelos conseguidos en diseño. En el flujo de implementación se deben implementar la arquitectura, las clases y los subsistemas, integrándolos para conseguir el sistema final. Además hay que implementar el sistema de navegación definido y los prototipos abstractos diseñados.

Una tarea esencial a tener en cuenta en este flujo es decidir en qué entorno habrá que realizar la implementación. Algunas veces este aspecto viene definido en la especificación de requisitos pero en otros casos no se sabe en qué lenguaje se trabajará hasta que no se ha diseñado el sistema.

Una vez definido el entorno y con las especificidades del diseño, la tarea de la implementación se convierte prácticamente en traducir a código lo que se ha diseñado. Así por ejemplo, del diseño básico se extrae información sobre la arquitectura del sistema, los sistemas de almacenamiento a usar y la estructura de la base de datos que se le asociará. Con el diseño de la interfaz abstracta y el diseño de la navegación se puede conocer qué pantallas hay que realizar, los eventos que la afectan, quién puede trabajar con ellas y cómo se puede acceder y salir de cada una de estas pantallas. Sí que habría que aplicar técnicas para hacer el diseño visual de estas pantallas, pero nada más.

Además, como en el flujo de diseño se propone el uso de patrones de diseño estándar, podemos saber y aplicar técnicas conocidas para conseguir el código asociado a estos patrones.

Este flujo de trabajo es uno de los menos desarrollados aún. Hasta el punto de que aún no se han definido las actividades a realizar.

6.3.6. Pruebas

Uno de los aspectos no contemplados en un elevado número de propuestas metodológicas es el que tiene relación con la definición de las pruebas que se han de realizar a un sistema para garantizar que el mismo responde a la definición de requisitos que para él se hizo. Aunque la definición concreta de las pruebas que se deben aplicar a un sistema deberá concretarse en el momento de realizar su diseño, toda propuesta metodológica debe definir las posibles estrategias a seguir, las técnicas a utilizar para realizarlas y el momento de hacer uso de cada una de ellas.

En el marco de la propuesta para el desarrollo de sistemas de información global que se está presentando, en el flujo de pruebas se debe elaborar, diseñar e implementar el plan de pruebas. Este plan de pruebas no solo debe recoger las pruebas a realizar, además debe indicar el orden de realización. Una vez realizado, hay que ejecutar el plan de pruebas y elaborar una memoria de resultados del mismo.

Igual que el flujo anterior, el flujo de pruebas tampoco está muy desarrollado aún. La idea es que se ofrezca también las actividades a desarrollar y los productos a obtener para conseguir ese plan de pruebas y el resultado de su ejecución.

6.4. Estado actual, puntos abiertos y trabajos publicados

La propuesta metodológica en la que se está trabajando está aún en sus comienzos. Nuestro trabajo comenzó realizando un estudio de la situación actual. Para ello, contactamos con personas que se estaban moviendo en el tema asistiendo a congresos específicos en este ámbito y analizando los trabajos publicados.

Una vez conocida la necesidad, se comenzó a trabajar en definir esta propuesta. La primera definición de la propuesta [Escalona 2000a] era una visión general de necesidades basada principalmente en los trabajos de Rossi y Schwabe en OOHDM y del Proceso Unificado.

Siguiendo el orden lógico, comenzamos con el primer flujo de trabajo, la especificación de requisitos. Para definirlo, estudiamos propuestas como las enunciadas de Durán o de Robertson. Así ha surgido una primera definición de fase. Esta propuesta de definición de requisitos ha dado origen a algunos trabajos que han sido enviados a diferentes congresos.

Actualmente estamos cerrando la fase de especificación y hemos comenzado con la fase de análisis en la que estamos definiendo las actividades y técnicas a aplicar.

Es necesario comentar que actualmente estamos trabajando conjuntamente con la Universidad de A Coruña de manera que la metodología pueda complementarse con los trabajos que ellos están realizando en temas de arquitectura [Brisaboa 2001]. Estamos usando y basándonos en trabajos paralelos dentro del propio grupo MADEIRA como son los trabajos en definición de interfaz [González 2001], de búsquedas eficientes [Gasca 2001] o de definición de arquitecturas [Cordero 2000].

También hemos visto desde el principio la necesidad de no definir una propuesta que al final no tenga aplicabilidad en el entorno industrial. Por ello, se están desarrollando trabajos a través de la organización FIDETIA (www.fidetia.us.es) en dos proyectos en los que estamos aplicando y mejorando nuestras propuestas. Estos proyectos cubren ámbitos tanto en empresas públicas, como es el caso del Instituto Andaluz de Patrimonio Histórico, como en empresas privadas, como es el caso de Sadiel.

Actualmente en estos entornos empresariales hemos puesto en práctica exclusivamente la fase de especificación. Los resultados nos han servido para mejorar la propuesta y acercarla más a la realidad. En definitiva hemos observado que la propuesta ha sido bastante bien acogida tanto por los clientes como por el resto de programadores, analistas y diseñadores del sistema.

Nuestro objetivo es el de terminar de definir nuestra metodología en cada uno de sus flujos de trabajos y seguir teniendo esta colaboración con las empresas para acercar la propuesta a los entornos profesionales.

Otro punto que actualmente se está realizando es el desarrollo de una herramienta CASE que ayude al seguimiento de la propuesta y valide los modelos realizados.

Para finalizar con este apartado, recogemos a continuación las publicaciones que actualmente tenemos en el tema. Algunas de ellas coinciden con las recogidas en la bibliografía pues han sido tomadas como referencia para la realización de este trabajo.

Publicaciones en revistas

✍✍M.J. Escalona, J.Torres, M.Mejías. **"Aplicación de los sistemas de tratamiento de bibliotecas digitales al Sistema de Información del Patrimonio Histórico Andaluz".** Boletín Trimestral del Instituto Andaluz de Patrimonio Histórico, Sevilla, Septiembre 2000.

Centrándonos en las bibliotecas digitales, un tipo concreto de aplicación son los sistemas para el tratamiento de bibliotecas digitales. Este trabajo propone desarrollar un sistemas para el tratamiento de bibliotecas digitales a un caso concreto para el que se definen las necesidades a cubrir: el sistema de información del patrimonio histórico de Andalucía.

✍✍J.M. Cordero, M.J. Escalona, J. Torres, M.Mejías, R.M. Gasca. **"Aplicación de los sistemas de tratamiento de bibliotecas digitales a la gestión de Patrimonio Histórico".** Número monográfico. Estudios Turísticos Nº 146. Pags. 37-47. Instituto de Estudios Turísticos. Madrid, Diciembre 2000.

Este trabajo se orienta en el ámbito de las bibliotecas digitales como caso concreto de sistema de información global. Es un trabajo conjunto en el que se plantea el uso de una arquitectura basada en la tecnología XML para desarrollar una biblioteca digital del patrimonio histórico de Andalucía. Fue seleccionado por esta revista tras su publicación en el congreso Turitec.

Publicaciones en congresos y reports internos

✍✍ M.J. Escalona, J.Torres, M.Mejías **"Propuesta de metodología para el desarrollo de sistemas para el tratamiento de bibliotecas digitales"**, Report Interno, 2-2000 del Departamento de Lenguajes y sistemas Informáticos. Universidad de Sevilla. Sevilla, Junio 2000.

Este trabajo fue el primero que publicamos. Se trata de un documento en el que se hizo una primera aproximación global de la propuesta basada fundamentalmente en las propuestas del Proceso Unificado y OOHDM.

✍✍J.M. Cordero, M.J. Escalona, J. Torres, M.Mejías, R.M. Gasca. **"Aplicación de los sistemas de tratamiento de bibliotecas digitales a la gestión de Patrimonio Histórico".** Congreso de Turismo y Tecnologías de la Información y las Comunicaciones: Nuevas Tecnologías y Patrimonio. Madrid, Octubre 2000.

Este trabajo se orienta en el ámbito de las bibliotecas digitales como caso concreto de sistema de información global. Es un trabajo conjunto en el que se plantea el uso de una arquitectura basada en la tecnología XML para desarrollar una biblioteca digital del patrimonio histórico de Andalucía.

✍✍ M.J. Escalona, M.Mejías, J.Torres. **"Aproximación metodológica al desarrollo de sistemas para el tratamiento de bibliotecas digitales"**. V Jornadas de Ingeniería del Software. Valladolid, Noviembre 2000.

La aproximación metodológica que se presenta en este trabajo es un resumen de la propuesta que actualmente se está desarrollando. Ya no está tan influenciada por el Proceso Unificado y OOHDM y es mucho más concreta en muchos aspectos. Está centrada exclusivamente en las bibliotecas digitales pero sirvió como base para el planteamiento de la propuesta general que se está elaborando actualmente.

✍✍ J.Torres, M.Mejías, M.J. Escalona, J.A. Ortega, JM. Cordero. **"Diseño del Modelo Navegacional para Sistemas de Tratamiento en Bibliotecas Digitales"** I Jornadas de Bibliotecas Digitales. Valladolid, Noviembre 2000.

Este trabajo ya se centra con más detalle en un aspecto concreto, el de la navegación. En este artículo se presentaron las técnicas para diseñar de forma adecuada las necesidades de navegación en un sistema para el tratamiento de bibliotecas digitales. Las ideas que se presentaron en este artículo eran lo suficientemente genéricas como para ser extendidas a los sistemas de información global.

✍✍ M.J. Escalona, M.Mejías, J.Torres, J.M. Cordero, M.González. **"Aplicación Integrada de la Biblioteca Digital del Patrimonio Histórico Andaluz".** I Jornadas de Bibliotecas Digitales. Valladolid, Noviembre 2000.

Este trabajo presentó la aplicación de nuestra propuesta metodológica para el desarrollo de sistemas basados en bibliotecas digitales a un caso concreto y real. Se pretendía aplicarla al desarrollo de un sistema que integrase toda la información sobre el patrimonio histórico de Andalucía.

✍✍ M. Mejías, M.J. Escalona, J.Torres. **"Captura de requisitos navegacionales y de interfaz abstracta para los sistemas de tratamiento de bibliotecas digitales"**. IV Jornadas Científicas en Tecnologías de la Información. Cádiz, Noviembre 2000.

Siguiendo el camino lógico, el primer flujo de trabajo con el que trabajamos fue la captura de requisitos. Este artículo presentó la propuesta que planteábamos para capturar y definir los requisitos de navegación e interfaz en los sistemas para el tratamiento de bibliotecas digitales.

✍✍ M.J. Escalona, M.Mejías, J.Torres, A.Reina. **"Propuesta metodológica para el desarrollo de sistemas para el tratamiento de bibliotecas digitales".** I Jornadas Dolmen. Sevilla, Junio 2001.

El grupo de investigación en el que participamos, se mueve dentro del grupo Dolmen. Este artículo es una visión general de la propuesta que hemos presentado en este trabajo de investigación. Aunque allí la presentamos en el marco de las bibliotecas digitales, actualmente ha evolucionado hacia los sistemas de información global.

✍✍ M.González, M.Mejías, M.J. Escalona, R.Martínez, J.A. Ortega. **"Interacción con los Usuarios en bibliotecas digitales".** I Jornadas Dolmen. Sevilla, Junio 2001.

Este artículo, también presentado en las jornadas del grupo de investigación, trata de justificar la importancia que la interacción tiene en los sistemas para el tratamiento de bibliotecas digitales y la necesidad de contar con ellos en el proceso de desarrollo.

✍✍ R.M. Gasca, M.J.Escalona, J.A. Ortega, M.Mejías, J.Torres.**"Aplicación de la programación con restricciones a la elaboración automática de itinerarios culturales"**. Congreso de Turismo y Tecnologías de la Información y las Comunicaciones: Nuevas Tecnologías y Patrimonio. Madrid, Octubre 2001.

Este es un trabajo conjunto con el grupo de trabajo que estudia los sistemas de recuperación de información dentro del grupo de investigación. Aplicando la programación con restricciones a la definición de requisitos que nosotros hicimos sobre el sistema de información del Instituto Andaluz de Patrimonio Histórico, se define un programa para generar automáticamente itinerarios culturales a través de internet.

✍✍M.J. Escalona, M. Mejías, J.Torres, A. Reina Quintero, J.M. Cordero**. "Requisitos de almacenamiento de información e identificación de actores para una biblioteca digital de bienes muebles".** II Jornadas de bibliotecas digitales. Ciudad Real, Noviembre 2001.

Este artículo, mucho más reciente que los anteriores, aplica las propuestas que se han presentado en el apartado 6.3.2 para la definición y descripción de actores del sistema a un problema real. Este problema real es el de los bienes muebles que hemos venido analizando en este trabajo.

✍✍N.R. Brisaboa, M.J. Escalona, M. Mejías, A.S. Places, F.J. Rodríguez, J. Torres. **"Sistema de consulta vía Web? para el Instituto Andaluz de Patrimonio Histórico".** II Jornadas de bibliotecas digitales. Ciudad Real, Noviembre 2001.

El artículo que presentamos aquí, fue el resultado de un trabajo conjunto realizado con la Universidad de A Coruña. El trabajo se basó en aplicar la metodología que estamos desarrollando y la arquitectura para bibliotecas digitales que ellos están planteando para estudiar su compatibilidad en un sistema real, el del Instituto Andaluz de Patrimonio Histórico. Actualmente estos trabajos de colaboración se siguen realizando.

✍✍M.J. Escalona, M.Mejías, J.Torres. **"Getting requirements in web information systems".** Fourteenth International Conference "Software & Systems Engineering & their Applications". Paris, Diciembre 2001.

Este es el último trabajo que hemos publicado. En el se presenta el flujo de trabajo de especificación de requisitos tal y como se ha presentado en este documento.

7. Revisión Bibliográfica

[Bieber 1998] M. Bieber, R. Galnares and Q. Lu. *Web enginering and flexible hypermedia.* The 2nd Workshop on Adaptative Hypertext and Hypermedia, Hypertext 1998.

[Blaha 1988] M.R. Blaha. *Relational database design using a object-oriented methodology*. Commun, ACM, Vol 31, Nº4, 1988, pp. 414-427

[Booch 1999] G. Booch, J. Rumbaugh, I. Jacobson. *Unified Modeling Language User Guide.* Ed. Addison-Wesley, 1999.

[Brisaboa 1999] N. Brisaboa. M.J. Durán, C. Lalín, J.R. López, J.R. Paramá, .R. Penabad, A.S. Places, *Arquitectura para Bibliotecas Virtuales*, 1999.

[Brisaboa 2001] N.R. Brisaboa, M.J. Escalona, M. Mejías, A.S. Places, F.J. Rodríguez, J.Torres. *Sistema de consulta vía Web? para el Instituto Andaluz de Patrimonio Histórico*. II Jornadas de bibliotecas digitales. Ciudad Real, Noviembre 2001.

[Caneiro 1994] L.M.F. Caneiro, M.H.Coffin, D.D. Coewan, C.J.P.L. Lucena. *ADVcHARTS A Visual Formalism for Higgly Interactive Systems.* in M.D. Harrison, C.Johnson, eds, Software Engineering in Human-Computer Interaction, Cambridfe University Press, 1994.

[Ceri 1999] S. Ceri, P.Fraternali and S. Paraboschi. *Desing Principles for Data-Intensive Web Sites.* SIMOG Record, 28: 84-89, 03 1999.

?

?

[Ceri 2001] S. Ceri, P. Fraternali, A. Bongio. *Web Modeling Language (WebML): a modeling language for designing Web sities.* First international workshop on Web-Oriented Software Technology. Valencia, Junio 2001.

[Cockburn 1997] A. Cockburn. *Structuring Use Cases with Goals. Journal of Object-Oriented Programming*, 1997.

[Coleman 1992] D.Coleman, F.Hayes, S.Bear. *Introducing Objectcharts or How to use Statecarts in Object-Oriented Design.* IEEE Transaction on software Engineering, 18(1), January, 1992.

[Conallen 1999a] J. Conallen. *Building Web Applications with UML*. Addison Wesley 1999.

[Conallen 1999b] J. Conallen. *UML Extension for Web Applications 0.91*. Disponible en http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm

[Cordero 2000] J.M. Cordero, M.J. Escalona, J. Torres, M.Mejías, R.M. Gasca. *Aplicación de los sistemas de tratamiento de bibliotecas digitales a la gestión de Patrimonio Histórico***.** Congreso de Turismo y Tecnologías de la Información y las Comunicaciones: Nuevas Tecnologías y Patrimonio. Madrid, Octubre 2000.

[Cowan 1995] D.D.Cowan, D.J.P.Lucena, *Abstract Data Views. An Interface Specification Concept to Enyhance Desingn for Reuse.* IEEE Transactions on Software Engineering. 18(1), 9-18, January 1995.

[De Troyer 1997] O.M.F. De Troyer, C.J. Leune. *WSDM: A User Centered Design Method for Web Sites*. Tilburg University, Infolab. 1997

[Durán 2000] A. Durán. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información.* Tesis Doctoral. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla. Septiembre 2000.

[Escalona 2000a] M.J. Escalona, J.Torres, M.Mejías. *Propuesta de metodología para el desarrollo de sistemas para el tratamiento de bibliotecas digitales*, Report Interno, 2-2000 del Departamento de Lenguajes y sistemas Informáticos. Universidad de Sevilla. Sevilla, Junio 2000.

[Escalona 2000b] M.J. Escalona, J.Torres, M.Mejías. *Aplicación de los sistemas de tratamiento de bibliotecas digitales al Sistema de Información del Patrimonio Histórico Andaluz*. Boletín Trimestral del Instituto Andaluz de Patrimonio Histórico, Sevilla, Septiembre 2000.

[Escalona 2000c] M.J. Escalona, M.Mejías, J.Torres. *Aproximación metodológica al desarrollo de sistemas para el tratamiento de bibliotecas digitales*. V Jornadas de Ingeniería del Software. Valladolid, Noviembre 2000.

[Escalona 2000d] M.J. Escalona, M.Mejías, J.Torres, J.M. Cordero, M.González. *Aplicación Integrada de la Biblioteca Digital del Patrimonio Histórico Andaluz*. I Jornadas de Bibliotecas Digitales. Valladolid, Noviembre 2000.

[Escalona 2001a] M.J. Escalona, M.Mejías, J.Torres, A.Reina. *Propuesta metodológica para el desarrollo de sistemas para el tratamiento de bibliotecas digitales*. I Jornadas Dolmen. Sevilla, Junio 2001.

[Escalona 2001b] M.J. Escalona, M. Mejías, J.Torres, A. Reina Quintero, J.M. Cordero**.** *Requisitos de almacenamiento de información e identificación de actores para una biblioteca digital de bienes muebles***.** II Jornadas de bibliotecas digitales. Ciudad Real, Noviembre 2001.

[Firesmith 1997] D.G.Firesmith. *Uses Cases: The Pros and Cons.* 1997.

[Fong 1995] J. Fong. *Mapping extended entity relationship model to object modeling techique*. SIGMOD Record, Vol. 24, Nº3, Septiembre 1995, pp. 18-22.

[Garzoto 1993] F. Garzoto, D.Schwae and P.Paolini *HDM-A Model Based Approach to Hypermedia Aplication Design*. ACM Trnasactions on Information System, 11 (1), Jan 1993, pp 1-26.

[Gasca 2001] R.M. Gasca, M.J.Escalona, J.A. Ortega, M.Mejías, J.Torres. *Aplicación de la programación con restricciones a la elaboración automática de itinerarios culturales*. Congreso de Turismo y Tecnologías de la Información y las Comunicaciones: Nuevas Tecnologías y Patrimonio. Madrid, Octubre 2001.

[Greco 1998] D. Greco, M. Eskelinen, C. Funckhouser, M.Luesebrink, J.Rosenberg *Actual & Potential Hypertext & Hypermedia.* ACM Digital Library. June 20-24, 1998.

[Gómez 2001] J. Gómez, C. Cachero and O.Pastor. *Extending a conceptual Modelling Approach to Web Application Design.* First international workshop on Web-Oriented Software Technology. Valencia, Junio 2001.

[González 2001] M. González, M. Mejías, M.J. Escalona, R.Martínez, J.A. Ortega. *Interacción con los Usuarios en bibliotecas digitales*. I Jornadas Dolmen. Sevilla, Junio 2001.

[Harrison 1993] W. Harrison, H. Ossher. *Subjects-Oriented Programming (a critique of pure objects)*. In Proceeding of the Conference on Object-Oriented Programming Systems, Languajes and Applications (OOPSLA'93), pp. 411-428, Washington, September 1993.

[Hennicker 2001] R. Hennicker, N. Koch. *A UML-based Methology for hypermedia Design*. First international workshop on Web-Oriented Software Technology. Valencia, Junio 2001.

[Hoch 1999] M. Hoch, D. Schwabe. *Group interaction in a surround screen environment*. Inst. for Visual Media ZKM, Karlsruhe, Germany. IEEE Computer Animation, 1999

[IBM OOTC 1997] IBM OOTC. *Developing Object Oriented Software*. IBM Object Oriented Technology Center. Prentice-Hall 1997

[IEEE 1993] *IEEE Recommended Practice for Software Requirements Specif*ications. IEEE/ANSI Standard 830-1993, Institute of Electronical and Electronics Engineers, 1993

[Izakowirz 1995] T. Izakowitz, E.Stohr, P. Balasubramaniam: *RMM:A metodology for structured hypermedia design*. Comm. Of ACM, October 1995, pp.34-35

[Jacobson 1995] I. Jacobson. *Modeling with use cases-Formalizing use-case modelling*. Journal of Object-Oriented Programming, June 1995

[Jacobson 1999] I. Jacobson, G. Booch, J. Rumbaugh. *The Unifed Software Development Process*. Ed. Addison-Wesley, 1999.

[Knapp 1997] A. Knapp, N. Koch, M. Wirsing. & others. *EPK-fix: Methods and Tools for engineering electronic product catalogues*. Interactive Distributed Multimedia Systems and Telecommunication Services, Lecture Notes in computer Science 1309, pages 1999-209. Springer-Verlag Berlin-Heidelberg, September 1997.

[Lange 1995] D.B. Lange. *An Object-Oriented Design Approach for Devloping Hipermedia Information Systems.* Research Report RT00112, IBM Research, Tokyo Research Laboratory, Japón, 1995.

[Lee 1998] H. Lee, C. Lee, C. Yoo. *A Scenario-based objec-oriented methodology for developing hypermedia information systems*. Procesings of 31st Annual Conferecne on Systems Science. Eds. Sprague R.

[Lesk 1997] M. Lesk. *Practical Libraries: Books, Bytes & Bucks.* Morgan Kaufmann. 1997.

[Liddle 2001a] S.W. Liddle, D.W. Embley, S.N. Woodfiel. *A seamless model for Object-oriented systens development.* First international workshop on Web-Oriented Software Technology. Valencia, Junio 2001.

[Liddle 2001b] S.W. Liddle, D.W. Embley, S.N. Woodfiel. *An Active, Object-Oriented, Model-Equivalent Programming Language.* First international workshop on Web-Oriented Software Technology. Valencia, Junio 2001.

[Maier 1997] C. Maier, L. Mandel. *YAON- a Static Diagram Technique for the Development of Electronic Product Catalogue* Proceedings of the Symposium on Software Technology, SADIO, August 1997.

[Mandel 2000] L. Mandel, A.Helmerich, L.A. Olsina, G.Rossi, M.Wirsing, N.Koch. *Hyper-UML. Specification and modeling of multimedia an Hypermedia Applications in Dystributed systems*. Agosto 2000.

[MAP 1995] MAP. *Metodología de Planificación y Desarrollo de Sistemas de Información. MÉTICA Versión 2.1: Guía de Referencia*. Tecnos/ Ministerio para las Administraciones Públicas 1995.

[Martínez 1997] J.M. Martínez, J.R.Hilera, J. Martínez, J.A. Gutiérrez. *Orientación a Objetos en la Documentación Hipermedia*. Universidad de Alcalá de Henares, Departamento de Ciencias de la Computación.

[Mecca 1999] G. Mecca, P. Atzeni, V. Crescenzi. *The ARANEUS Guide to Web-Site Development*. Technical Report, Universidad de Roma, 03 1999.

[Mejías 2000] M. Mejías, M.J. Escalona, J.Torres. *Captura de requisitos navegacionales y de interfaz abstracta para los sistemas de tratamiento de bibliotecas digitales*. IV Jornadas Científicas en Tecnologías de la Información. Cádiz, Noviembre 2000.

[Nanard 1995] J. Nanard, M. Nanard. *Hypertext design enviroments and the hypertext design process*. Communication of the ACM, August 1995. Vol 38(8), 49-56. 1995.

[Olsina 1998] L. Olsina. *Building a Web-based information system applying the hypermedia flexible process modeling strategy*. 1st International workshop on Hypermedia Development, Hypertext 1998.

[Pastor 1992] O. Pastor, F. Hayes and S. Bear. *OASIS: An object-Oriented Specification Languaje*. CAiSE'92 International Conference, volume 593 of LNCS, pages 948-363. Springer-Velag, 1992.

[Pastor 1997] O. Pastor, E.Insfran, V. Pelechano, J.Romero and J. Merseguer. OO-METHOD: *An OO Software Production Enviroment Combining Conventional and Forma Methods*. CAiSE'97. International Conference on Advanced Information Systems, 1997.

[Piattini 1996] M.G. Piattini, J.A. Calvo-Manzano, J.Cervera y L.Fernández. *Análisis y Diseño detallado de Aplicación Informáticas de Gestión*. ra-ma 1996

[Raghavan 1994] S. Raghavan, G. Zelesnik y G. Ford. *Lectures Notes of Requirements Elicitacitation*. Educational Materials CMU/SEI-94-EM-10 1994

[Rossi 1996] G. Rossi. *An Object Oriented Method for Designing Hipermedia Applications.* PHD Thesis, Departamento de Informática, PUC-Rio, Brazil, 1996.

[Rossi 1997] G. Rossi, D. Schwabe, A. Garrido: *Design Reuse in Hipermedia Applications Development*. Proceedings of ACM International Conference on Hypertext (Hypertext`97), Southampton, April 7-11, 1997. ACM Press.

[Rumbaugh 1991] J. Rumbaugh. *Modelado y Diseño Orientado a Objetos.* Ed. Prentice Hall, 1991.

[Rumbaugh 1999] J. Rumbaugh, I. Jacobson, G. Booch. *The Unified Modeling Language Reference Manual*. Ed. Addison-Wesley, 1999.

[Scheneider 1998] G. Schneider y J.P. Winteers. *Applying Use Cases: a ractical Guide.* Addison Wesley 1998.

[Schwabe 1995] D.Schwabe, G.Rossi. *The Object-Oriented Hipermedia Design Model*. Communications of the ACM, 38(8), August, 1995, pp.45-46

[Thomson 1998] J. Thomson, J. Greer and J. Cooke. *Algorithmically detectable design patterns for hypermedia collections.* Workshop on Hypermedia development Process, Methods and Models. Hypermedia 1998.

[Torres 2000] J.Torres, M.Mejías, M.J. Escalona, J.A. Ortega, J.M. Cordero. *Diseño del Modelo Navegacional para Sistemas de Tratamiento en Bibliotecas Digitales*. I Jornadas de Bibliotecas Digitales. Valladolid, Noviembre 2000.

[Robertson 2000] J. Robertson and S. Robertson. VOLERE. *Requirements Specification Templates*. 2000

[Vinze 2001] A.S. Vinze. *Globalization of the Business Enterprise Effects and Impacs of Information Technology.* Departament of Business Analysis. Texas A&M University.

[Wirfs-Brock 1990] R. Wirfs-Brock, L. Wilkerson & P. Wiener. *Desingning Object-oriented Software.* Prentice-Hall 1990.

[Witten 1996] I. Witten, C. Nevill-Manning, S. Cunningham. *Building a Digital Library for Computer Science Research: Technical Issues*. Proceedings of the 19th Australesian Computer Science Conference. 1996.

[WWW 2000] World Wide Web Consortium. www.w3c.org, 2000.

[Yourdon 1989] Yourdon E. *Modern Structured Analysis*. Prentice-Hall 1989.

8. Foros relacionados

Debido a la generalidad de los sistemas de información global, existen muchos foros y páginas de interés para estos sistemas.

A nivel de grupos de trabajo son especialmente interesantes:

- A nivel nacional
  - El grupo de trabajo del proyecto Dolmen. Es un grupo de trabajo formado por un total de seis universidades españolas entre ellas las de Sevilla.
  - El grupo de trabajo de la red de Bibliotecas Digitales. En esta también participan diversas universidades españolas, como la de Valencia, la Coruña y la de Sevilla.

- A nivel internacional
  - A nivel internacional sería impensable nombrar a todos los grupos de trabajo que existen que trabajan en el entorno de la ingeniería del software orientados a sistemas de información global. Pero sería destacable el comentar el grupo de trabajo que ha surgido en torno al desarrollo de metodologías para la web. En este grupo, que aún no tiene un nombre definido, se encuentras personas como Gustavo Rossi o Daniel Schwabe, autores de OOHDM; Oscar Pastor, autor de OO-Method; Franca Garzo, autora de RMM; Luis Olsina, autor de HFPM; u

Olga de Troyer, autora de WSDM. Este grupo se reunieron por última vez en Valencia y van a publicar un libro con los resultados.

- Otro grupo a nivel internacional es el de Hyper-UML. Hyper-UML es un proyecto conjunto entre Alemania, con personas como Nora Koch o Luis Mander, y Sudamérica, con participantes como Gustavo Rossi o Daniel Schwabe. Este grupo pretende desarrollar una metodología para la web, partiendo de los trabajos anteriores de estos autores.

Con respecto a los congresos en los que se tratan aspectos de este tema son también múltiples. En la mayoría de congresos en los que se trata ingeniería del software, aspectos multimedia o aspectos de la web son interesantes.

Por resaltar algunos tenemos:

- A nivel nacional
  - Las jornadas anuales de Ingeniería del Software y Bases de Datos.
  - Las jornadas anuales de Bibliotecas Digitales
  - El congreso anual Turitec. Que aunque no es puramente técnico, resulta muy interesante por el acercamiento que ofrece entre los investigadores informáticos y los temáticos.

- A nivel internacional
  - Las jornadas anuales *Software & Systems Engineering and their applications*. La última celebración se realizará en Diciembre de 2001 en Paris.
  - Las jornadas anuales *Interoperability in Digital Libraries.* La última celebración ha sido a principios de Septiembre de 2001 en Alemania.
  - La conferencia anual International Conference on Enterprise Information Systems. Su próxima celebración será en Abril de 2002 en Ciudad Real.

Por último, también podríamos destacar numerosas revistas que publican temas de esta índole.

- D-Lib Magazine. www.dlib.org
- Programming and Computer Software. www.wkap.nl/journalhome.htm
- Revista de Computación y Sistemas. www.cic.ipn.mx/~Cys/
- Advances in Engineering Software. www.elsevier.nl/locate/advengsoft
- ACM Transactions on Information System www.acm.org/tois
- ACM Transactions on Software Engineering and Methodologies www.acm.org/tosem
- Advances in Engineering Software www.elsevier.nl/locate/advengsoft

- Annuals of software Engineering www.baltzer.nl/ansoft
- IEEE Computing in Science & Engineering computer.org/cise
- IEEE Computer computer.org/computer
- IEEE Multimedia. computer.org/multimedia
- IEEE Transactions on Computer computer.org/tc
- IEEE Transactions on Multimedia www.ieee.org/organizations/tab/tmm.html
- IEEE Transactions on Software Enginering. computer.org/multimedia
- Information software technology www.elsevier.com/locate/infsoft
- Information systems www.elsevier.com/locate/infsys
- Interacting with computer www.elsevier.com/locate/intcom
- International journal on Digital Libraries link.springer.de/link/service/journals/00799/index.html
- International journal on software engineering and knowledge engineering www.ksi.edu/ijsk.html
- Engineering and technology management www.elsevier.com/locate/jengtecman
- The journal of System and Software www.elsevier.com/locate/jss
- Multimedia Systems link.springer.de/link/service/journals/00530/index.html
- Requirements Engineering link.springer.de/link/service/journals/00766/index.html
- Science of Computer Programming www.elsevier.com.locate.scico