Ingeniería de la web y patrones de diseño

Este libro nace con el objetivo de dar una panorámica global sobre el proceso de construcción de sistemas y aplicaciones hipermedia y, en particular, de los orientados a la web, haciendo especial hincapié en aspectos relevantes como el diseño, la reutilización de conocimiento, la calidad, la seguridad y, por supuesto, en el uso de patrones de diseño.

Pretende ser un libro académico y, al mismo tiempo, práctico por lo que cada uno de los capítulos incluye distintas experiencias de aplicación y gran número de ejemplos destinados a conseguir que el lector comprenda los beneficios de cada una de las aproximaciones presentadas.

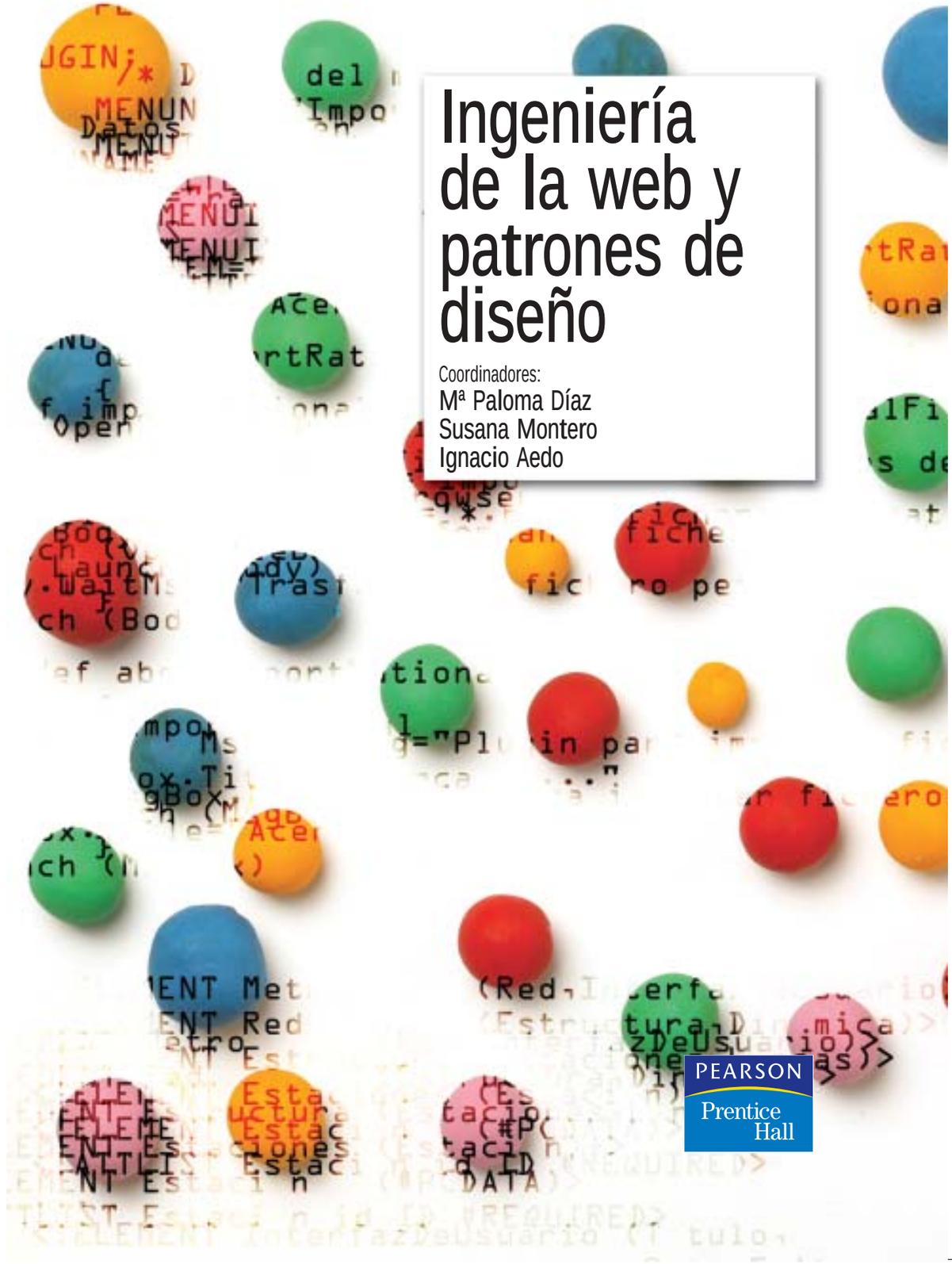
> *The authors of this book have provided us with a well-argued and valuable resource. It has the potential to help scholars enhance their understanding, and practitioners to improve the quality of the systems being developers. The authors have shown significant vision in drawing together a set of material, where each chapter provides a thread in a rich tapestry. (*Del prólogo, *David Lowe, University of Technology, Sydney)*

Este libro ha sido publicado gracias a la ayuda aportada por el proyecto “Sistema de Ayuda para la Construcción de Aplicaciones Hipermedia/Web Basado en el Uso de Patrones de Diseño” financiado por la Dirección General de Investigación de la Comunidad Autónoma de Madrid y el Fondo Social Europeo con referencia 07T/0024/2003.

PEARSON Educación
www.pearsoneducacion.com

ISBN 84–205–4609–7
9 788420 546094

Ingeniería de la web y patrones de diseño



Foreword (Prólogo original)

Web Engineering is an emerging discipline which, as eloquently described by the authors, has to address some significant complexities in the systems being developed. Much of this complexity arises out of the interplay between the diverse areas which form the basis of Web systems: information systems; psychology; graphic design; marketting; amongst others. Web Engineering is fundamentally about integrating these areas into an effective approach to the development of quality applications. Developers have a rather fundamental choice to make: adopt an *ad hoc* approach to development with little control over quality and extreme difficulty in ensuring that the system addresses actual needs; or adopt a well-structured web engineering approach which is based on sound principles and captures best-practice in developing effective solutions.

If we are indeed to adopt an quality Web Engineering approach, then a key element is the management of knowledge and the sharing of "best practice"! This in turn raises the question of how we capture, represent and share this knowledge. This book highlights a key mechanism for achieving this - patterns.

The concept of patterns has existed for a considerable period, with strong roots in the work of the architect Christopher Alexander in the late 1960's and early 1970's. During the 1990's this work began to be adapted to Software Engineering, emerging strongly in the well known "Gang of Four" (named for the four authors) book "*Design Patterns: Elements of Reusable Object-Oriented Software*". Patterns are often naively viewed just a template or an example solution, but in reality are much more than this. Consider the following:

> *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*. – ChristopherAlexander

> *A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It de-*

005 PROLOGO 2/6/05 12:02 Página xviii

> *scribes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples.* – "Design Patterns: Elements of Reusable Object-Oriented Software"

In other words, a pattern needs to explicitly form the link between a problem and the nature of a "best practice" solution to that problem. So, why is this relevant to Web Engineering? Well, returning to my earlier comments, Web development is highly multidiscplinary, evolutionary, and volatile. Indeed Web systems are deeply interwoven into (and indeed mutually constituted with) the organisations which they support. This mutual constitution means that the organisational needs drive the form which a Web system will take, but the form of the Web system is in turn also increasingly moulding the nature of the organisations which they support.

This interdependance is often handled in Web development through the extensive use of prototypes to explore the potential impact which a proposed design may have on the nature of the need being addressed. In some senses, this can be viewed as a form of reverse impact analysis, where we are trying to understand how a solution impacts on the problem, rather than vice versa.

So, within this type of development environment, not only does managing knowledge about these systems become more critical —something which patterns inherently support— but this increased interdependance between the problem space (i.e. the organisation problems needing to be addressed) and the solution space (the systems being constructed) becomes more complex to handle.

If we naively proceed with development, then we risk constructing Web systems which ultimately lead to changes in the needs which they were actually trying to address. Patterns become even more crucial in this context. Not only do they help us identify potential solutions to an identified problem, but used with more subtlety they can help us look at a proposed solution (i.e. Web design) and see what sort of problem (i.e. business process? Stakeholder need?) that solution might really be addressing or changing! In other words, patterns can be used in a much richer way than has often been the case in the past - to help developers and clients to understand the impacts which a proposed design of a Web system might have on their business!

It is in this area that lies the incredible strength and value of this book. As you read through this book, I would encourage you to think (as I have) not only about how the approaches described can help you in constructing Web-based solutions to identified needs (i.e. a conventional software engineering view of development) but how this rich collection of inovative work can help you in understanding the complex interplay between client needs and system designs.

The authors of this book have provided us with a well-argued and valuable resource. It has the potential to help scholars enhance their understanding, and practitioners to improve the quality of the systems being developers. The authors have shown significant vision in drawing together a set of material, where each chapter provides a thread in a rich tapestry. I hope you enjoy this book and benefit from it as much as I have.

David Lowe
Sydney, Australia
February 2005

Prólogo (Traducción)

La Ingeniería de la Web es una disciplina emergente que, como elocuentemente describen los autores de este libro, tiene que hacer frente a importantes dificultades en los sistemas en desarrollo. Muchas de estas dificultades surgen de la interacción entre las diversas áreas que forman la base de los sistemas Web: los sistemas de información; la psicología; el diseño gráfico y la mercadotecnia, entre otras. La Ingeniería de la Web consiste fundamentalmente en la integración de estas áreas en un enfoque efectivo para el desarrollo de aplicaciones de calidad. Los desarrolladores tienen que hacer una elección absolutamente fundamental: adoptar en el desarrollo un enfoque *ad hoc* con un escaso control sobre la calidad y una extrema dificultad para garantizar que el sistema cumple con las necesidades reales, o adoptar un enfoque bien estructurado de ingeniería de la web que esté basado en sólidos principios y que capture las mejores prácticas en el desarrollo de soluciones efectivas.

Si realmente vamos a adoptar un enfoque de la ingeniería de la web que sea de calidad, la gestión del conocimiento y la compartición de buenas prácticas son aspectos clave. Esto a su vez nos lleva a la cuestión de cómo capturamos, representamos y compartimos ese conocimiento. Este libro aborda un mecanismo básico para conseguir este objetivo: los patrones.

El concepto de patrones existe desde hace tiempo y tiene profundas raíces en el trabajo que el arquitecto Cristopher Alexander desarrolló a finales de los 60 y comienzos de los 70. En los 90 este trabajo comenzó a adaptarse a la Ingeniería del Software, surgiendo con fuerza en el famoso libro *Design Patterns: Elements of Reusable Object-Oriented Software*, más conocido como "*Gang of four*" (llamado así por sus cuatro autores). Los patrones son a menudo vistos ingenuamente como simples formatos o ejemplos de solución, pero en realidad son mucho más que eso. Considérese lo siguiente:

> *"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y, después, describe la solución a ese problema de tal forma que se pue-*

005 PROLOGO 2/6/05 12:02 Página xx

> *de utilizar esa solución un millón de veces más, sin hacerlo dos veces de la misma manera".* -- - Christopher Alexander
>
> *Un patrón de diseño nombra, motiva y explica de forma sistemática un diseño general que afronta un problema de diseño recurrente en los sistemas orientados a objetos. Describe el problema, la solución, cuándo aplicar la solución y sus consecuencias. También ofrece pistas para la implementación y ejemplos.* – "Design Patterns: Elements of Reusable Object-Oriented Software"

En otras palabras, el patrón necesita establecer explícitamente un enlace entre un problema y la esencia de una "mejor práctica" que solucione ese problema. Pero, ¿por qué es esto relevante en la Ingeniería de la Web? Bien, retornando a mis comentarios anteriores, el desarrollo Web es profundamente multidisciplinar, evolutivo y volátil. De hecho, los sistemas Web están estrechamente entremezclados con las organizaciones que sustentan, e incluso ambos están mutuamente constituidos. Esta mutua constitución significa que las organizaciones necesitan dirigir la forma que tiene el sistema Web, pero a su vez la forma de esos sistemas Web está moldeando cada vez más la naturaleza de las propias organizaciones.

Esta interdependencia es a menudo manejada en el desarrollo Web a través del uso extensivo de prototipos para explorar el potencial impacto que una propuesta de diseño podría tener en la naturaleza de la necesidad que está siendo abordada. En cierto sentido, esta práctica puede verse como una forma de análisis inverso de impacto, en el que tratamos de entender cómo una solución afecta a un problema en vez de a la inversa.

Así, en este tipo de entorno de desarrollo, no sólo la gestión del conocimiento sobre esos sistemas se convierte en más crítica – algo que los patrones inherentemente soportan- sino que la creciente interdependencia entre el espacio del problema (es decir, los problemas de la organización que tienen que ser solventados) y el espacio de la solución (es decir, los sistemas en construcción) se vuelve más difícil de manejar.

Si procedemos a la ligera con el desarrollo, entonces corremos el riesgo de construir sistemas Web que acaben dando lugar a cambios en las necesidades que se suponía que iban a solventar. Los patrones se vuelven aún más cruciales en este contexto. No sólo nos ayudan a identificar posibles soluciones a un problema identificado, sino que utilizados de una manera más sutil pueden ayudarnos a observar una solución propuesta (i.e. un diseño Web) y ver qué clase de problema (¿proceso de negocio? ¿necesidad de los destinatarios?) debería solventar o cambiar esa solución. Dicho de otra forma, los patrones pueden utilizarse de manera más rica de lo que se ha venido haciendo habitualmente: ¡para ayudar a los desarrolladores y a sus clientes a entender el impacto que una propuesta de diseño de un sistema Web puede tener en su propio negocio!

Es en esta área en la que reside la increíble solidez y valor de este libro. Según vaya leyéndolo, le animo a pensar (como yo he hecho) no sólo en cómo los enfoques descritos pueden ayudarle a construir soluciones basadas en Web a necesidades ya identificadas (i.e. una visión convencional en la ingeniería del software de lo que es el desarrollo) sino en cómo esta rica colección de trabajos innovadores puede ayudarle a comprender la compleja interacción entre las necesidades de los clientes y los diseños del sistema.

005 PROLOGO 2/6/05 12:02 Página xxi

Los autores de este libro nos han proporcionado un recurso bien argumentado y valioso. Tiene el potencial de ayudar a los estudiantes a incrementar su compresión y a los profesionales a mejorar la calidad de los sistemas que desarrollan. Los autores han tenido gran visión al reunir una serie de materiales en los que cada capítulo constituye un hilo en un rico tapiz. Espero que disfrute este libro y se beneficie de él tanto como yo lo he hecho.

David Lowe
Sydney, Australia
Febrero 2005

Ingeniería de la web y patrones de diseño
Resúmenes

PARTE I: INTRODUCCIÓN

1 El desarrollo hipermedia y web como proceso de ingeniería

Ignacio Aedo, Paloma Díaz, Susana Montero, Manuel Castro

En este primer capítulo se profundiza en aspectos básicos del desarrollo sistemático de sistemas hipermedia y web o, lo que es lo mismo, en lo que habitualmente se conoce como ingeniería de la hipermedia/web, analizando aspectos genéricos y sin centrarse en ningún método de desarrollo concreto. Así, indica qué fases se deben tener en cuenta en dicho proceso de desarrollo para posteriormente abordar los diversos modelos de proceso que pueden aplicarse y describir las características propias del desarrollo de sistemas hipermedia que lo hacen distinto del desarrollo de sistemas de información y que deben ser tenidas en cuenta en métodos orientados a esta tecnología. Finaliza el capítulo con una breve panorámica de algunos métodos de desarrollo no incluidos en este libro, concluyendo con un resumen del objetivo y contenidos de este libro.

PARTE II: MÉTODOS DE DESARROLLO PARA LA WEB

2 Diseño de hipermedia y web con ADM

Paloma Díaz, Ignacio Aedo, Susana Montero

En este capítulo se presenta un método llamado "*Ariadne Development Method"* (ADM), que plantea un proceso sistemático, integrador e independiente de la plataforma de implementación para modelar y evaluar aplicaciones y sistemas hipermedia. Por un lado, el modelo de proceso es iterativo y centrado en el usuario con objeto de mejorar la *usabilidad* de las aplicaciones resultantes. Por otro, se establecen un conjunto de fases en las que se deben generar una serie de productos mediante los cuales se tienen en cuenta todas las características del sistema, ya sean de navegación, estructurales, de presentación, de interacción, de acceso o de funcionamiento. Tres aspectos clave de ADM son: incluir productos para diseñar las características propias de los contenidos multimedia, ofreciendo la posibilidad de especificar relaciones espacio-temporales; abordar el modelado de requisitos funcionales, tanto de los relacionados con las capacidades de navegación como de otros servicios complejos; y finalmente, soportar el modelado de usuarios y de políticas de acceso. ADM cuenta con una herramienta de automatización, AriadneTool, con la que se pueden generar prototipos a partir de los productos diseños.

3 Diseño de Aplicaciones Web con VisualWADE

Jaime Gómez, Antonio Párraga, Oscar Asensi, Jose Antonio Navarro

El desarrollo de aplicaciones web es una tarea compleja que requiere del uso de una amplia variedad de conocimientos de tecnología, organización y comunicación . Los sistemas de información basados en web son mucho más complejos que las aplicaciones tradicionales debido a que han de construirse sobre componentes tecnológicos que se encuentran en continua evolución, han de encajar en la infraestructura existente en la empresa y la interfaz de usuario debe de ofrecer un nivel de calidad hasta ahora no exigido. Este capítulo describe los fundamentos ingenieriles de VisualWADE, una herramienta CASE que automatiza la producción de interfaces de usuario de aplicaciones web. VisualWADE es una aproximación dirigida por el modelo (*model-driven*) que hace énfasis en el análisis de requisitos, el diseño de alto nivel, y el prototipado rápido. De esta forma una aplicación evoluciona de forma suave desde el primer prototipo al producto final y su mantenimiento es consecuencia natural del desarrollo. Conforme aparecen nuevos requisitos o cambios en los actuales, simplemente hace falta revisar el modelo conceptual, establecer los cambios y regenerar la implementación.

4 MIDAS: Una Aproximación Dirigida por Modelos para el Desarrollo Ágil de Sistemas de Información Web

Belén Vela, Paloma Cáceres, Valeria de Castro, Esperanza Marcos

En este capítulo se describe MIDAS, una metodología ágil dirigida por modelos para el desarrollo de Sistemas de Información Web (SIWs). MIDAS propone una arquitectura dirigida por modelos, basada en la propuesta MDA (*Model Driven Architecture*) del OMG, e integra técnicas procedentes de metodologías ágiles como *eXtreme Programming* y *Agile Modeling*.

La arquitectura de MIDAS propone modelar el SIW respecto a dos dimensiones ortogonales: por un lado, teniendo en cuenta los principales aspectos: contenido, hipertexto y comportamiento; y, por otro lado, respecto a los niveles MDA: modelos independientes de computación, modelos independientes de

plataforma y modelos específicos de plataforma. Además, MIDAS está basada en el uso de estándares para el desarrollo de SIW y propone usar UML como notación única para modelar todo el sistema.
Este trabajo se centra en los aspectos de: hipertexto, definiendo una aproximación orientada al usuario; y contenido, definiendo un proceso de desarrollo de una base de datos web para las tecnologías objeto-relacional y XML. Además se presentan las lecciones aprendidas de los diferentes casos de estudio desarrollados para validar la propuesta. Actualmente, se está aplicando la metodología a una aplicación real: el SIW para la gestión de imágenes médicas.

5 El Desarrollo Documental

José Luis Sierra, Alfredo Fernández-Valmayor, Baltasar Fernández-Manjón, Antonio Navarro

En este capítulo los autores describen una metodología de construcción de aplicaciones basada en la documentación marcada de las mismas: el desarrollo documental de aplicaciones ricas en contenidos. El principal factor de variabilidad de este tipo de aplicaciones, típicas en la web, viene dado por un cuerpo de contenidos sobre un determinado dominio, que es producido y mantenido por expertos en dicho dominio. El desarrollo documental se centra en la producción de (i) un conjunto de documentos que describen los aspectos más relevantes de la aplicación (v.g. los contenidos, así como otras características configurables de la misma, como, por ejemplo, diferentes propiedades de su interfaz de usuario), (ii) una gramática que caracteriza un lenguaje de marcado específico del dominio (LMED) que se utilizará para marcar descriptivamente dichos documentos, y (iii) un procesador para dicho LMED. La aplicación en sí se produce procesando los documentos que la describen con un procesador para el LMED. Los autores han aplicado satisfactoriamente este enfoque en la producción y mantenimiento de aplicaciones educativas y aplicaciones hipermedia, así como de sistemas basados en el conocimiento. En base a dichas experiencias se ha constatado que la factibilidad práctica del desarrollo documental precisa de mecanismos para la formulación incremental del LMED y para la construcción incremental de su procesador. Este hecho ha conducido a la formulación de (i) la Aproximación Documental al Desarrollo de Software (ADDS), un modelo de proceso para el desarrollo documental, (ii) la técnica de Provisión de LMEDs en ADDS (PADDS), una solución para la formulación incremental de LMEDs, y (iii) el modelo de Operacionalización en ADDS (OADDS), un modelo para la construcción incremental de procesadores para dichos LMEDs.

6 Desarrollo de Entornos Virtuales para Web

María-Isabel Sánchez-Segura[1], Angélica de Antonio[2], Gonzalo Méndez[2]

Los Entornos Virtuales constituyen un tipo de sistema altamente interactivo para cuyo desarrollo se hace necesario encontrar nuevas técnicas y formalismos especialmente adaptados, ya que los métodos y técnicas clásicos de la Ingeniería del Software han demostrado empíricamente no ser suficientes. En este capítulo, tras una introducción a los entornos virtuales, sus principales características y tipologías, y una revisión de las diferentes tecnologías involucradas en su construcción, se presenta una aproximación al proceso de desarrollo para este tipo de sistemas, el marco metodológico SENDA, profundizando en el conjunto de actividades y técnicas propuestas para la fase de análisis. Por último se hace una revisión de las principales áreas en que estos sistemas están teniendo acogida.

PARTE III: ASPECTOS FUNDAMENTALES EN EL DESARROLLO PARA LA WEB

7 Modelado de la Seguridad en Sistemas de Información Web

Daniel Sanz, Paloma Díaz, Ignacio Aedo

Este capítulo subraya la necesidad de considerar la seguridad durante todo el ciclo de desarrollo de sistemas hipermedia y web. Se centra en las medidas de seguridad de alto nivel, en concreto en el control de accesos, enfatizando la necesidad de emplear modelos adecuados para reflejar la política de accesos al sistema. También se pretende resaltar la importancia de integrar estos aspectos de modelado en el proceso de desarrollo, en lugar de postergarlos para etapas finales del ciclo de vida, lo que les reduciría a meras decisiones de implementación. Se propone un modelo de control de accesos para hipermedia denominado MARAH, y se ilustra su aplicación en el marco de un método de diseño, poniendo como ejemplo el diseño del sistema ARCE.

8 Calidad en el desarrollo de aplicaciones web: modelos, métodos de evaluación y métricas de calidad

María Dolores Lozano, Francisco Montero, Fco. Javier García, Pascual González

Este capítulo aborda el estudio de la calidad de las aplicaciones web, pretendo especial atención a uno de los factores de calidad más importantes en este tipo de entornos: la usabilidad. Para ello se analiza el concepto de calidad del software desde diferentes perspectivas y se describen los distintos factores que pueden ayudar a describir dicho concepto. Junto a lo anterior se introduce el concepto de modelo de calidad como elemento de referencia para evaluar la calidad de un sistema, describiéndose los distintos tipos de técnicas de evaluación propuestos hasta el momento. Por último, se aborda el estudio de la inclusión de la evaluación de la usabilidad dentro del proceso de desarrollo, algo esencial para garantizar la calidad de un producto software.

9 Ingeniería de la accesibilidad a la web

Julio Abascal, Myriam Arrue, Nestor Garay, Juan Miguel López, Markel Vigo

La mayor parte de la información y de las aplicaciones actualmente disponibles en la web presenta serias barreras de accesibilidad para determinados tipos de usuarios a causa del modo en que ha sido creada. Está demostrado que si los diseñadores siguen determinadas pautas de accesibilidad al crear sitios web, todas las personas, provistas de una interfaz adecuada a sus necesidades, pueden acceder a ellos sin mayores dificultades. Sin embargo, la aplicación de estas pautas de diseño no es fácil. Es necesario el uso de herramientas adecuadas que estén preferiblemente integradas en el *software* de desarrollo. Además, si se quiere obtener resultados fiables y duraderos, la metodología de trabajo debe incluir el diseño accesible como objetivo prioritario dentro del ciclo de vida de la tecnología web. Este capítulo describe los problemas de accesibilidad y cómo enfrentarlos, las herramientas de trabajo existentes y su integración en el proceso de diseño de aplicaciones web.

PARTE IV: PATRONES DE DISEÑO

10 Patrones e Interfaz de usuario

Pedro J. Molina

El campo del diseño de interfaces de usuario ha comenzado a sacar partido durante la última década a los patrones software. Tanto desde el punto de vista de toma de requisitos, modelado conceptual, diseño, implementación o mantenimiento es posible explotar los patrones como herramienta de ayuda en todas estas fases. Este capítulo presenta un recorrido en el uso de los patrones dentro del campo del diseño de interfaces de usuario.

11 Capitalizando experiencia de diseño en aplicaciones web

Matías Butti, Juan Danculovic, Gustavo Rossi

Durante el proceso de construcción de aplicaciones web existe un proceso importante previo a la implementación que es la etapa de diseño. Diseñar y documentar las decisiones de diseño de una manera sistemática conlleva a reducir la complejidad y el costo asociado del mantenimiento y la extensión de la aplicación en construcción.

Los métodos existentes abarcan todas las incumbencias de diseño con una estrategia sistemática, a través de un diseño conceptual o aplicativo, un diseño navegacional y un diseño visual. Gracias a no acoplar los métodos de diseño a una tecnología particular, la etapa de diseño no depende de la tecnología que se utilice en la posterior implementación.

Para cualquiera de las incumbencias, encapsular experiencia de diseño usando patrones es relevante para transferir al sistema en construcción soluciones ya probadas en diseños exitosos. Si bien existe un gran catalogo de patrones de diseño, en el artículo se presentan un conjunto de los mismos seleccionados por cuestiones didácticas: Landmarks, Novedades, Recomendación, Comunicación Push, Personalización de estructura, Espacio de búsqueda, Respuesta estructurada, Enlace tipado, Información bajo demanda. Estos patrones no son simplemente parte de un catalogo, sino que se aplican a diario en la construcción de aplicaciones web, por ejemplo en el desarrollo de una aplicación para el campus de una universidad como se presenta en el capítulo.

12 Integración de patrones en el proceso de diseño de aplicaciones hipermedia

Susana Montero, Paloma Díaz, Ignacio Aedo

Los métodos de diseño hipermedia asisten al diseñador a través de una serie de fases y productos. Sin embargo, un diseñador, tanto si es experto como no, puede llegar a tomar una solución inadecuada a un determinado problema de diseño que puede ser perjudicial para el usuario. Los patrones de diseño son ese mecanismo que puede ayudar a aliviar tanto el acometido de nuevos desarrollos como la dificultad en la toma de decisiones para la mejora de sistemas ya existentes, en tanto en cuanto puedan integrarse de forma eficiente con los métodos de diseño y sean accesibles de manera efectiva a los diferentes

integrantes del equipo de desarrollo. En este capítulo se presentan una serie de mecanismos dirigidos a agilizar la recuperación y aplicación de los patrones en conjunción con los métodos.

1 El desarrollo hipermedia y web como proceso de ingeniería

Ignacio Aedo, Paloma Díaz, Susana Montero y Manuel Castro[1]

Laboratorio DEI. Universidad Carlos III de Madrid
Avda. de la Universidad 30, 28911 Leganés
aedo@ia.uc3m.es, pdp@inf.uc3m.es, smontero@inf.uc3m.es
http://dei.inf.uc3m.es
[1]Departamento de Ingeniería Eléctrica, Electrónica y Control. UNED
C/ Juan del Rosal, 12, 28040 Madrid
mcastro@ieec.uned.es

1.1 Introducción

Los sistemas hipermedia organizan la información multimedia en una serie de unidades conceptuales, habitualmente denominadas nodos, que están relacionadas por medio de enlaces navegables que hacen posible la libre exploración del espacio de información por parte de los usuarios. La idea básica subyacente a este tipo de aplicaciones software es el uso de una organización asociativa de la información, similar a la que se emplea en la memoria a medio y largo plazo para relacionar los recuerdos [65], y tiene sus orígenes en Memex, el mágico escritorio que ideó Vannevar Bush en 1945 [13] para poder enlazar conceptos relacionados de distintas disciplinas con el fin que se pudiese mejorar la transferencia de conocimiento. En la década de los sesenta comienzan a surgir los primeros sistemas hipertexto, denominados así por incluir exclusivamente información textual y gráfica, tales como Augment/NLS desarrollado por Douglas Engelbart en 1963 [29] o HES (*Hypertext Editing System*) construido por Andrew van Dam y Ted Nelson en 1967. Desde entonces se han desarrollado muchas herramientas de construcción, sistemas aplicados a diversas áreas, métodos de desarrollo o modelos de referencia. La hipermedia se ha convertido en una tecnología madura de la que existe una base teórica y una establecida comunidad de investigadores que trabajan en aspectos de muy diversa índole: desde temas puramente técnicos, como puedan ser la definición de nuevas arquitecturas [2; 44; 62; 81], la aplicación de técnicas de ingeniería del software para producir sistemas de calidad [35; 41; 50; 71], la especificación de modelos de referencia que recojan las características y el funcionamiento de este tipo de sistemas [14; 22; 43; 46; 79], la integración eficiente de material multimedia [52] o el desarrollo de interfaces de usuario avanzadas [11; 74; 75]; hasta otros de carácter más literario como son la exploración de nuevos modelos de lectura y escritura para mejorar la comprensión de la información [82].

Desde el punto de vista de la aplicación, los sistemas web pueden considerarse como un subconjunto de los sistemas hipermedia, en concreto aquel que comprende las aplicaciones hipermedia que funcionan en el entorno específico de la web. De hecho, muchas características y funciones propias de los sistemas hipermedia han ido

llegando de forma paulatina al entorno web y algunas aún no se han incorporado de manera estándar. Así por ejemplo, la capacidad de sincronizar contenidos, básica para poder crear composiciones multimedia, ya propuesta en modelos de hipermedia como Amsterdam [46] empezó a plantearse en web con el desarrollo del lenguaje SMIL (*Synchronized Multimedia Integration Language*) [76], cuya primera recomendación se publica en junio de 1998. Otras características, como los enlaces n-arios introducidos en el modelo de referencia de Tompa en 1989, aún no tienen una equivalencia directa en la web. Es en este sentido en el que en este capítulo se considera que el concepto de sistema web está englobado en el de hipermedia, a sabiendas de que existen múltiples aspectos adicionales, tales como aquellos relacionados con los protocolos de comunicación o los servicios de bajo nivel ofrecidos por los servidores de información, que constituyen de por sí una interesante y activa área de investigación.

Una de las áreas de investigación que ha experimentado un gran auge en los últimos años es la del uso de un enfoque sistemático en el desarrollo de este tipo de sistemas y aplicaciones, o lo que es los mismo la ingeniería de la hipermedia y de la web. Así, uno de los principales problemas a que ha tenido que hacer frente esta tecnología es que los sistemas hipermedia, que empiezan a aparecer a mediados de los años sesenta como se comentó anteriormente, son previos a cualquier teoría al respecto, ya sea modelo, meta-modelo o método de desarrollo, lo cual dio lugar a una situación de incomunicación entre sistemas que alertó a la comunidad de investigadores en este área, culminando con el primer taller (*workshop*) sobre estandarización celebrado en Gaithesburg en el año 1990. Además, con el advenimiento de los sistemas web el problema se agravó. La mayor parte de las organizaciones y compañías se ven forzadas a poner en marcha sistemas hipermedia, fundamentalmente en forma de sitios web, de una forma precipitada, lo que hace que se sumerjan en esta tecnología sin un sólido conocimiento de sus fundamentos teóricos. Ello conlleva que, en la mayor parte de los casos, no se siga un enfoque propio de la ingeniería, hasta tal punto que algunos autores ya han identificado lo que sería la **`crisis del software hipermedia`** [55], una reminiscencia de proceso de software inmaduro. Por todo ello es obligado pensar que se necesita una ingeniería de la hipermedia que guíe los procesos de construcción de este tipo de aplicaciones y, en particular, una ingeniería de la web que ayude al a los equipos de desarrollo durante toda la construcción de este tipo concreto de sistema hipermedia.

Es evidente que el desarrollo de sistemas hipermedia y web a gran escala debe llevarse a cabo siguiendo un proceso sistemático y bien definido [4] y no puede realizarse de forma artesanal, básicamente por las mismas razones que se aplican métodos de ingeniería de software en otras disciplinas. En este sentido es remarcable el común error en que muchos desarrolladores de sistemas hipermedia, particularmente de sistemas web, incurren al considerar que no existen modelos ni métodos por lo que, en el mejor de los casos, se conforman con utilizar modelos de otras tecnologías, especialmente UML (*Unified Modeling Language*) [9], para tratar de documentar un sistema que suele realizarse sin ninguna fase previa de conceptualización. Este enfoque es absolutamente erróneo, puesto que el uso de los métodos y las técnicas existentes para diseño de hipermedia permiten no sólo mejorar la calidad, la reusabilidad y la facilidad de mantenimiento de los sistemas desarrollados sino también beneficiarse de la experiencia y el conocimiento de otros

diseñadores. Así por ejemplo, el modelo HDM (*Hypermedia Design Model*) [41] propone una estructura jerarquizada que facilita la navegación por el hiperespacio; el método RMM (*Relationship Management Methodology*)[49], junto con su versión extendida ERMM (*Extended Relationship Management Methodology*) [50], proponen una serie de herramientas de navegación condicionales bastante útiles y OOHDM (*Object-Oriented Hypermedia Design Model*) comienza a incorporar ciertos patrones de diseño que se han empleado de forma recurrente y con buenos resultados en el desarrollo y operación de este tipo de sistemas (véase el capítulo 11).

Si bien los métodos y técnicas propios de la ingeniería del software no son directamente aplicables al desarrollo de sistemas hipermedia, puesto que no recogen los aspectos estéticos y cognitivos propios de los sistemas hipermedia [59] que, además, se desarrollan siguiendo un proceso muy iterativo y progresivo [56], tampoco la experiencia acumulada en este campo a lo largo de todos estos años es desdeñable, antes bien puede ayudar considerablemente como se apunta en [56]. De hecho no hay más que observar que la mayor parte de los modelos de referencia o de diseño y de los métodos de desarrollo suelen hacer uso de técnicas, abstracciones y notaciones propias de otros ámbitos, con especial énfasis en el modelo relacional [17] y la orientación a objetos [9; 69].

Una de las razones fundamentales que hacen que sea preciso utilizar métodos y técnicas específicamente desarrollados para esta tecnología, entre otras, es la necesidad de contar con mecanismos que permitan modelar [25;41]:

1. Sofisticadas estructuras de navegación, algunas de las cuáles pueden ser efímeras o adaptables a las necesidades y preferencias del usuario.
2. Comportamientos interactivos y reacciones ante determinados eventos, ya sean iniciados por el usuario (v.g. cuando un usuario pincha sobre un vídeo que se está reproduciendo, éste se para y vuelve al primer fotograma) o no (v.g. cuando dos objetos que se están moviendo de forma aleatoria por el espacio de visualización chocan suena un pitido).
3. Interfaces con aplicaciones externas, tales como bases de datos, servicios web u otras aplicaciones hipermedia.
4. Composiciones multimedia en las que hay que armonizar contenidos que se presentan en distintas dimensiones, de tal manera que se dé lugar a una presentación usable al mismo tiempo que estéticamente adecuada a las necesidades y preferencias de sus usuarios.
5. Restricciones de acceso que hagan posible indicar cómo distintos tipos de personas van a poder hacer uso del mismo sistema hipermedia de acuerdo con sus necesidades y responsabilidades [3; 26]. Estas restricciones deben ser expresadas en función de términos y abstracciones propias del dominio de la hipermedia, de tal forma que se pueda indicar quién puede ver un enlace o modificar un nodo.

En conclusión, si bien la experiencia en ingeniería de software es útil, fundamentalmente por la disciplina que impone al desarrollo y las capacidades de abstracción y modelado que permite adquirir, se requieren métodos que estén fundamentados en elementos y conceptos propios del dominio de la hipermedia, como puedan ser el nodo, el enlace o las relaciones temporales entre contenidos. Todo ello refuerza la idea de que se necesita un proceso de ingeniería para la hipermedia, y en particular para el desarrollo de sistemas y aplicaciones web que

hagan uso de métodos y técnicas específicos que ayuden al desarrollador en su labor de construcción.

Además, hay que tener en cuenta que el proceso de construcción de sistemas hipermedia, en especial de aquellos que están orientados a la web, se caracteriza por el poco tiempo de que se dispone para conseguir que el producto esté operativo. No obstante, esta premura no puede dar lugar a sistemas de baja calidad que contrarresten los beneficios de una rápida salida al mercado. Para conseguir sobrevivir a tan extremos tiempos de desarrollo, los diseñadores más experimentados suelen recurrir a soluciones que han utilizado en otros proyectos, adaptándolas de tal modo que les permitan resolver los problemas propios del desarrollo en curso. En este sentido, los patrones de diseño se perfilan como una técnica que favorece la reutilización de conocimiento de diseño, ya que hacen posible la expresión de una solución ya probada a un problema recurrente. Este enfoque no sólo facilita el trabajo de desarrollo a los diseñadores expertos sino, y muy especialmente, de los noveles. El uso de patrones no sólo ayuda a reducir el tiempo de desarrollo sino que también, y lo que es más importante, permite dedicar el potencial intelectual de los diseñadores a pensar soluciones a nuevos problemas o aplicaciones realmente novedosas en vez de a tratar de reinventar la rueda. En el terreno de la ingeniería hipermedia y de la web existen muchas propuestas de patrones que se pueden utilizar durante el desarrollo de aplicaciones y sistemas web como son los catálogos de Van Duyne, Landay y Hong [80] o el desarrollado a iniciativa del grupo de interés de ACM-SIGWEB en colaboración con la Universidad Suiza-Italiana (*University of Italian Switzerland*) que se puede encontrar en el Repositorio de Patrones de Diseño Hipermedia (http://www.designpattern.lu.unisi.ch). Para finalizar cabe resaltar que dichos patrones son sólo soluciones a problemas recurrentes pero no constituyen per se métodos de desarrollo y, además, su utilización no es inmediata ni sencilla, como se discutirá en el capítulo 12.

En este primer capítulo se profundizará en aspectos básicos del desarrollo sistemático de sistemas hipermedia y web o, lo que es lo mismo, en lo que habitualmente se conoce como ingeniería de la hipermedia/web, analizando aspectos genéricos y sin centrarse en ningún método de desarrollo concreto. Así, la sección 1.2 se dedica a las fases propias de dicho proceso de desarrollo para en la sección 1.3 abordar los diversos modelos de proceso que pueden aplicarse. En la sección 1.4 se describen características propias del desarrollo de sistemas hipermedia que lo hacen distinto del desarrollo de sistemas de información y que deben ser tenidas en cuenta en métodos orientados a esta tecnología. Una breve panorámica de algunos métodos de desarrollo no incluidos en este libro es presentada en la sección 1.5 y, finalmente, se concluye el mismo con un resumen del objetivo y contenidos de este libro.

1.2 Las fases del ciclo de vida del desarrollo hipermedia

Antes de pasar a estudiar en detalle las peculiaridades del proceso de desarrollo de las aplicaciones hipermedia, resulta conveniente repasar algunos conceptos propios de la ingeniería del software con el fin de establecer un vocabulario mínimo que se utilizará a lo largo de este capítulo. Con este objeto se ha recurrido al Glosario de

Terminología de Ingeniería del Software de IEEE (*IEEE Standard Glossary of Software Engineering Terminology*) [48].

En primer lugar es preciso distinguir entre el **ciclo de vida del desarrollo software** y el **modelo de proceso** del ciclo de vida, usualmente denominado modelo de proceso. El ciclo de vida en sí incluye, de manera genérica, una serie de fases entre las que se encuentran el análisis, el diseño, la implementación y las pruebas o la instalación, pero en ningún caso implica que estas fases tengan que realizarse siguiendo una determinada secuencia. El modelo de proceso es el que establece una forma de trabajo concreta en función del paradigma adoptado (por ejemplo, en cascada, iterativo, en espiral o incremental).

El modelo de proceso, a su vez, tampoco es un **método de desarrollo**. Mientras el modelo establece una secuenciación en las fases del desarrollo, el método propone de forma detallada qué actividades deben llevarse a cabo durante cada una de las fases y qué productos o entidades de diseño deben generarse. El método también ofrece principios básicos, guías o consejos para producir un software de mayor calidad. Así, existen distintos modelos de proceso que determinan cómo llevar a cabo las distintas fases del desarrollo y, a su vez, para cada modelo de proceso existirán diversos métodos que indicarán qué hacer en cada fase así como las interacciones que existen entre las distintas fases.

La adopción de métodos de ingeniería durante el proceso de desarrollo de los productos software, independientemente del tipo de producto del que se trate, hace posible una aplicación sistemática de conocimiento científico con objeto de construir soluciones efectivas y eficientes a un problema dado [8]. Así, la utilización de dicho enfoque en el ámbito de las aplicaciones interactivas estará orientado a [55]:

- entender los objetivos y requisitos del producto a desarrollar,
- diseñar interfaces adecuadas y estructurar la información de acuerdo con las necesidades del usuario,
- incorporar mecanismos que posibiliten un uso efectivo del producto por parte del usuario final,
- gestionar el proceso de desarrollo de manera eficiente,
- emplear métricas que recojan aspectos del desarrollo con los que se pueda controlar su progreso,
- documentar aspectos relevantes del desarrollo y
- llevar a cabo un desarrollo que asegure que la aplicación va a ser fácil de mantener.

El objetivo fundamental será pues obtener productos de calidad y llevar a cabo un proceso de desarrollo de calidad. Por un lado, un **producto hipermedia de calidad** será relevante, completo, correcto, usable y útil. Un **proceso de desarrollo hipermedia de calidad** garantizará la productividad, la reutilización, la facilidad de mantenimiento, la gestión del proceso y la medición, tanto del producto como del propio proceso.

El proceso de desarrollo de aplicaciones hipermedia conlleva la realización de una serie de actividades, independientemente de la secuencia que se siga en las mismas, entre las que se cuentan el estudio de la factibilidad, el análisis, el diseño, la producción, la evaluación y el mantenimiento.

Estudio de factibilidad

El objeto de esta fase es determinar si un determinado producto software es factible, tanto desde un punto de vista técnico como práctico. En el caso de un sistema interactivo, es importante analizar la aceptación que éste podría tener antes de embarcarse en un complejo y costoso desarrollo. Aspectos como la adecuación social del sistema o su utilidad práctica pueden tenerse en cuenta a este efecto [60].

Durante el estudio de factibilidad se deben considerar todos aquellos factores que podrían afectar al desarrollo y al éxito del producto final, entre los que se cuentan las limitaciones económicas, técnicas, de recursos, funcionales o cognitivas [55].

Análisis

El análisis es una actividad orientada a estudiar las características o requisitos de un producto software. Teniendo en cuenta que los productos hipermedia son sistemas interactivos orientados a dar soporte a unas determinadas tareas, de las que tiene conocimiento el usuario, una de las actividades básicas de la etapa de análisis es, precisamente, el **análisis de las tareas**. Así, se deberá saber qué actividades se van a poder llevar a cabo, en qué secuencia, con qué limitaciones, qué opciones existen para cada tarea, etc.

Además, es preciso conocer las **características de los usuarios** que pueden afectar al diseño. Por ejemplo, hay que tener en cuenta las capacidades o discapacidades físicas o cognitivas, la diversidad cultural, la edad, el sexo o cualquier otro parámetro que pueda influir en la forma de presentar la información o en la manera en que se va a producir la interacción.

También habrá que analizar el **entorno de operación**, en el cual se va a hacer uso del producto a desarrollar. Así habrá que establecer si existen limitaciones o estándares a cumplir, si se va a dar soporte al uso de diferentes plataformas de acceso o incluso las características físicas del entorno. Por ejemplo, a la hora de diseñar la interfaz de usuario de un punto urbano de información será necesario tener en cuenta características ambientales, como la luminosidad o el ruido.

El objetivo último de esta fase es entender qué debe hacer el producto y generar una especificación o pliego de requisitos en la que se recojan todas las características funcionales, no funcionales y de usabilidad de la aplicación.

Diseño

El proceso de diseño consiste en convertir una especificación de lo qué el producto debe hacer en una propuesta, más o menos formal, de cómo debe hacerlo. Durante el diseño se van a especificar, en consecuencia, aspectos tales como: cómo se va a estructurar la información, cómo se va a presentar la información al usuario, cómo funciona la aplicación, cómo va a poder interactuar el usuario con el producto en cada momento, o cómo se va a poder acceder al producto aplicando ciertas reglas de accesibilidad y de seguridad. Con este fin es preciso hacer uso de un **modelo de diseño** que permita traducir las necesidades en productos más o menos formales que puedan ser discutidos por parte de los diversos miembros del equipo de desarrollo.

El diseño es un proceso típicamente creativo y abstracto que se ve restringido por limitaciones técnicas, cognitivas y no técnicas [55] que deberían haberse recogido

durante el análisis y que deben tenerse en cuenta durante el diseño de determinadas partes del producto:

- **Restricciones técnicas**. Si bien se suele decir que el diseño debe centrarse sólo en el cómo hacer las cosas, no se puede hacer un buen diseño si se ignoran ciertas restricciones de la plataforma técnica, como por ejemplo la forma en que se va a distribuir el producto o las características típicas de la plataforma de uso. Así, si la mayor parte de los usuarios acceden a un curso virtual haciendo uso de un módem, no es lógico plantear un diseño en el que se incluya la videoconferencia como principal medio de comunicación.
- **Restricciones cognitivas.** Los usuarios tienen una características físicas (agudeza visual, capacidad auditiva, etc.) y cognitivas (memoria a corto y largo plazo, capacidad de razonamiento y de resolución de problemas) que deben ser respetadas si se pretende construir un sistema utilizable [27]. Por ejemplo, si se muestran varias animaciones simultáneamente, no se puede esperar que el usuario atienda a su contenido.
- **Restricciones no técnicas.** Existen otras restricciones a considerar como son aspectos legales, de seguridad o de autenticidad. Así, no se puede permitir por ejemplo que un curso virtual deje a sus usuarios acceder a información que viola la legalidad vigente (v.g. datos personales de otros usuarios).

El diseño es, por lo tanto, una compleja actividad en la que se tienen que amalgamar distintos requisitos bajo una misma perspectiva. A la hora de realizar el diseño pueden emplearse diferentes niveles de abstracción y producir diagramas o especificaciones conceptuales o incluso prototipos que ayuden a establecer la solución más adecuada para cada problema. Los capítulos 2, 3, 4, 5 y 11 presentan diversos métodos y técnicas de diseño para productos hipermedia y web.

Producción

A la hora de producir o generar una aplicación hipermedia existen diversas actividades a tener en cuenta, como se muestra en la Figura 1.1.

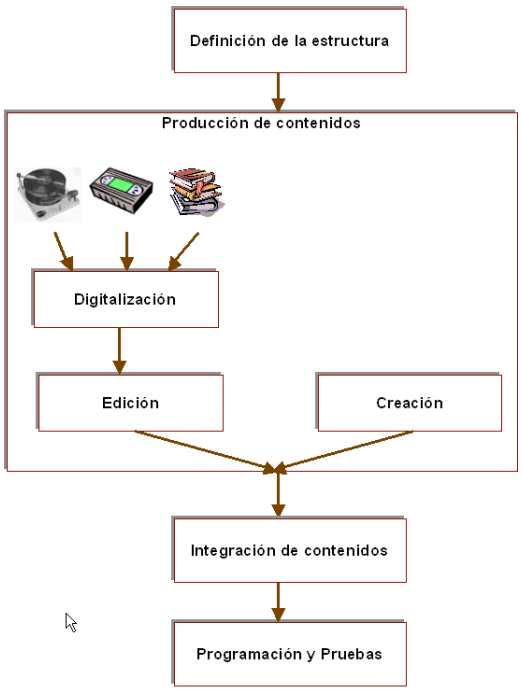


Figura 1.1 Proceso de producción de aplicaciones hipermedia, ampliación de la propuesta de [55]

En primer lugar es preciso establecer la organización de la aplicación, es decir identificar de qué conceptos se va a hablar o qué pantallas se van a incluir. Una aplicación multimedia está formada por una serie de **nodos** o unidades indivisibles de presentación en las que se incluyen varios contenidos. Así, cada ventana o cada marco puede considerarse un nodo. Durante la etapa de definición de la estructura se identificarían los nodos y la forma en que éstos se secuencian. Además, se determinarán los contenidos que se van a incluir en cada una de esos nodos, ya sean textos, gráficos, dibujos, animaciones, simulaciones, mundos virtuales, música, sonido o cualquier otro tipo de contenido que se considere de utilidad para conseguir los objetivos del producto.

El siguiente paso consiste en producir esos contenidos en un formato procesable por el ordenador. Es posible que se cuente con un guión que describa cómo debe ser el contenido y que haya que generar éste, pero también es posible que se quieran utilizar contenidos que existen pero que no están disponibles en formato no digital. Así, es muy frecuente que se cuente con vídeos o sonidos analógicos y con textos e imágenes en papel que se desean incluir. En este caso el paso previo es la digitalización utilizando el hardware y software adecuado. Una vez obtenido un archivo en formato digital se pasará a retocar el contenido multimedia editándolo con herramientas software al efecto. Este proceso, ya sea de creación o de digitalización y edición, no sólo requiere software y hardware específico, sobre el que se puede consultar en los temas específicos de este libro, sino también de la participación de

especialistas en diseño multimedia capaces de editar y generar contenidos de calidad. Cuestión aparte, muy importante en la producción web, es la gestión de los derechos de autor de cada una de las piezas de información que incorpora el sitio debiendo mantenerse el origen y los derechos asociados de cada una de esas piezas.

Una vez que se tiene la estructura y el contenido, basta con integrar los contenidos en esa estructura y programar aquellos procesos que sean necesarios. Como resultado de esta fase se habrá construido un sistema o un prototipo, dependiendo del modelo de proceso elegido y del estado en que se encuentre el desarrollo.

Evaluación

La evaluación tiene como misión primordial asegurar que las aplicaciones se han diseñado teniendo en cuenta las necesidades de sus usuarios finales, y no sólo las de los desarrolladores. Este proceso va a proporcionar información que permita comprobar si los mecanismos de interacción se han diseñado correctamente, detectando aquellas deficiencias que haya que solventar o proponiendo mejoras.

Es muy importante tener presente que en ningún caso la evaluación es una parte de la etapa de pruebas y depuración, ya que los errores que se buscan con la evaluación están relacionados con la forma de interactuar con el producto desarrollado y tienen su origen en una mala comprensión o interpretación de la forma en que el usuario se comunica con el sistema, y no con posibles fallos o errores en el código. Más detalladamente, este tema se tratará en el capítulo 8.

Mantenimiento

El mantenimiento del software consiste en modificar el producto o un componente del mismo una vez que ya se ha entregado, ya sea para corregir errores, mejorar el funcionamiento o alguna otra característica o para adaptarlo a cambios en el entorno [48].

Se estima que en sistemas de información una gran parte del esfuerzo del desarrollo se invierte en mantenimiento, cosa que en el caso de los sistemas hipermedia puede variar, especialmente cuando se trata de productos cerrados. En algunas ocasiones la aplicación hipermedia se concibe como un CD que una vez generado no va a evolucionar. Pero salvo en esos casos específicos, las aplicaciones multimedia deben mantenerse como cualquier otro tipo de producto software, un mantenimiento especialmente acusado en el caso de los sitios web.

1.3 Modelos de proceso para el desarrollo multimedia

Como se ha comentado con anterioridad, el modelo de proceso establece una forma de trabajo concreta en función del paradigma adoptado. Entre los paradigmas clásicos de modelo de proceso encontramos los siguientes [1]:

- El **`modelo en cascada`**, que exige finalizar una fase antes de poder pasar a la siguiente.
- El **`modelo incremental`**, en el que se van construyendo versiones del sistema, cada una de las cuales hace frente a un subconjunto de los requisitos.

- El **`modelo evolutivo`**, que está orientado a conseguir una versión rápida y flexible del producto, susceptible de ser modificada ante una variación en los requisitos.
- El **`modelo en espiral`**, en el que se hace un desarrollo iterativo basado en el análisis de riesgos.
- El **`modelo basado en transformaciones`**, que hace uso de herramientas CASE (*Computer Aided Software Engineering*) capaces de transformar automáticamente la salida de cada fase en entrada de la siguiente.
- El **`modelo basado en el uso de prototipos`** que ofrece una aproximación iterativa en la que se emplean prototipos para involucrar al usuario.
- El **`modelo en estrella`,** que consiste en realizar el proceso de evaluación después de cada etapa de desarrollo pudiendo volver a cualquiera de las etapas como resultado de ese proceso de evaluación.

El uso de prototipos y su evaluación por parte del usuario o de evaluadores expertos es básico en los sistemas interactivos, como los sistemas hipermedia. Por ello, de los modelos de proceso descritos anteriormente los que serían aplicables son los que conllevan un desarrollo iterativo. A continuación se presenta el modelo de la Ingeniería de la Usabilidad como un ejemplo de modelo de proceso orientado a sistemas interactivos, y en particular a sistemas web.

1.3.1 El modelo de proceso de la Ingeniería de la Usabilidad

El modelo de Mayhew [57] presentado en la Figura 1.2 es un ejemplo de proceso que se puede seguir en un desarrollo orientado a la web. En este modelo hay tres grandes fases: el análisis de los requisitos; el diseño, evaluación y desarrollo; y, por último, la de instalación.

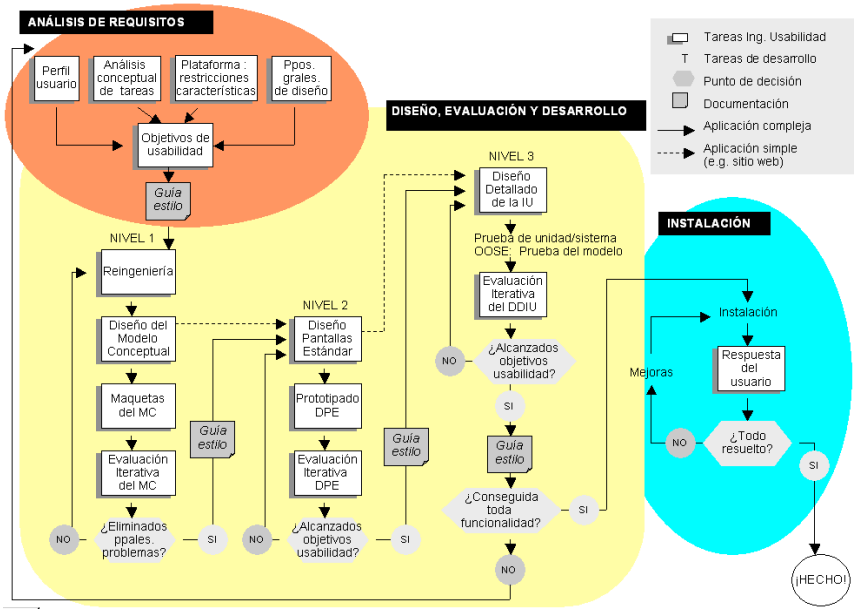


Figura 1.2 Ciclo de vida de la Ingeniería de la Usabilidad [57]

El análisis de requisitos se inicia estableciendo el perfil de los usuarios del sistema y llevando a cabo el análisis conceptual de las tareas, la definición de las restricciones y necesidades de la plataforma de uso y de los principios de diseño a aplicar. Todo ello da lugar a una serie de objetivos de usabilidad que el sistema debe cumplir y que, junto las guías de estilo, conforman el conjunto estable de requisitos a considerar en el proceso de desarrollo.

En la segunda fase, la de diseño, evaluación y desarrollo, se definen tres niveles de desarrollo que se corresponden con distintos niveles de abstracción, cada uno de los cuales hace uso de maquetas y prototipos como artefactos destinados a ser evaluados. En el primer nivel, se realiza el diseño conceptual y se construyen maquetas que representan la interacción con el sistema y que se evalúan para comprobar que se han cubierto todos los aspectos identificados en la fase anterior. El segundo nivel se centra en el diseño de las ventanas y en la construcción de los prototipos que van a permitir garantizar de manera iterativa si los objetivos de usabilidad se cumplen. Por último, la interfaz se va refinando de manera iterativa hasta conseguir un sistema que incorpora todas las funcionalidades requeridas.

Una vez que el producto se ha desarrollado se pasa a la fase de instalación del mismo en el entorno de explotación en el que el usuario real interactuará con el producto. El objetivo principal de esta fase es detectar y corregir posibles problemas.

En la Figura 1.2 se representa mediante la flecha discontinua el camino a seguir para construir sistemas web, considerados por Mayhew como un tipo simple de sistema interactivo. Como se puede observar, este modelo de proceso se centra en el desarrollo de prototipos de alto nivel en los que se van incorporando y evaluando sucesivamente las funcionalidades del sistema.

1.3.2 Selección del modelo de proceso

A la hora de abordar el desarrollo de un producto multimedia concreto es preciso adoptar un modelo de proceso, ya sea diseño iterativo, ciclo de vida en estrella o cualquier otro que se considere adecuado.

Existen múltiples factores que influyen de manera decisiva en esta decisión, entre los que se cuentan los siguientes:

- **`Tiempo de desarrollo`**: la mayor parte de los productos hipermedia, y en especial aquellos que se desarrollan como sitios web, deben generarse en un tiempo récord, por lo que el ciclo de vida debe permitir crear un producto o un prototipo rápidamente [55], aunque sin olvidar la calidad del producto final.
- **`Tamaño de la aplicación`**: no es lo mismo desarrollar una pequeña aplicación, por ejemplo un disco compacto sobre la Antigua Roma que se va a entregar con una revista, que una aplicación de gran envergadura, como por ejemplo el sitio web de un banco electrónico. Cada tipo de aplicación requiere de un tipo de ciclo de vida. Así, mientras que para productos pequeños un enfoque muy formal y rígido puede ser engorroso e inútil, para productos a gran escala es imprescindible aplicar un proceso completamente sistematizado [55].

- **Características del equipo de desarrollo**: dependiendo de los conocimientos del equipo de desarrollo se podrá adoptar un modelo de proceso u otro. Así, si la mayor parte de los integrantes tienen un perfil técnico será mejor aplicar un modelo iterativo en el que la participación de otro tipo de personas (v.g. diseñadores gráficos o usuarios) pueda posponerse hasta que se pueda contar con ellos.

Además, no hay que olvidar que en desarrollos complejos y de larga duración estos factores suelen variar, por lo que la característica más importante a tener en cuenta del modelo de proceso es su flexibilidad y capacidad de adaptación al cambio. En este sentido, el ciclo de vida en estrella y el de la Ingeniería de la Usabilidad (véase la sección 1.3.1) proporcionan mayor flexibilidad, permitiendo adaptar el proceso de desarrollo a las necesidades de cada momento. Por otro lado, el desarrollo iterativo marca unas pautas a seguir que pueden resultar de utilidad cuando es preciso disciplinar a los miembros del equipo de desarrollo, requisito bastante habitual en equipos con poca experiencia y multidisciplinares.

1.4 Características del desarrollo de los sistemas hipermedia y web

Los sistemas hipermedia y web tienen características intrínsecas que hacen preciso el uso de enfoques distintos a los que se aplican en otras disciplinas como pueda ser el desarrollo orientado a objetos. Así, la primera característica singular que hay que destacar es que en el ciclo de desarrollo es preciso introducir una fase de evaluación que permita la construcción del sistema mediante un proceso iterativo, en el que en cada iteración se pueda analizar la utilidad y la usabilidad del sistema con el objetivo de que, por un lado, se cumplan las expectativas y, por otro, los usuarios sean capaces de utilizarlo de manera eficaz y eficiente.

Además, existen otros aspectos a tener en cuenta en el proceso de desarrollo de sistemas interactivos que afectan a la selección del modelo de proceso y que hacen que se precisen métodos específicos que tengan en cuenta las peculiaridades de este tipo de productos software, entre las que cabe destacar las que se describen a continuación:

- el desarrollo debe estar centrado en el usuario y en sus necesidades,
- el equipo de trabajo es pluridisciplinar y
- existen algunos requisitos de modelado específicos del desarrollo de sistemas hipermedia y web.

1.4.1 Desarrollo centrado en el usuario

Las aplicaciones hipermedia, como aplicaciones interactivas que son, tienen como objeto conseguir que el usuario pueda llevar a cabo alguna tarea de una manera eficiente y efectiva, ya sea aprender algo, comprar un producto, comunicarse con alguien o simplemente divertirse. Dicha tarea se realiza a través de un diálogo que se establece entre el usuario y la máquina, que se materializa a través de la interfaz de usuario.

La interfaz de usuario puede representarse por medio de un modelo general en el que se establecen las relaciones entre los principales agentes involucrados, los entornos y los procesos realizados por ambos.

La Figura 1.3 es una simplificación del modelo propuesto por Roy Rada en [66]. En ella, los agentes que participan en la interacción son dos: la persona y el ordenador, y el canal a través del cual se materializa el diálogo entre ambos es la interfaz. El conocimiento sobre la tarea a realizar lo tiene la persona, que es realmente quien sabe qué quiere hacer y cuál es la secuencia de pasos a seguir para conseguir su objetivo. Cuando una persona se pone delante de una máquina para completar una tarea, analiza las funciones que le ofrece el producto software y realiza acciones orientadas a conseguir su objetivo. A su vez, en el entorno de la máquina el soporte a dicha tarea está implementado de una determinada forma y cada interacción del usuario da lugar a un procesamiento que va a generar una salida específica que el usuario interpretará para decidir si el sistema funciona como él había presupuesto. En la Tabla 1.1 se muestra un ejemplo del proceso cognitivo que se produce cuando un usuario que nunca ha utilizado un procesador de textos intenta emplearlo para escribir una carta.

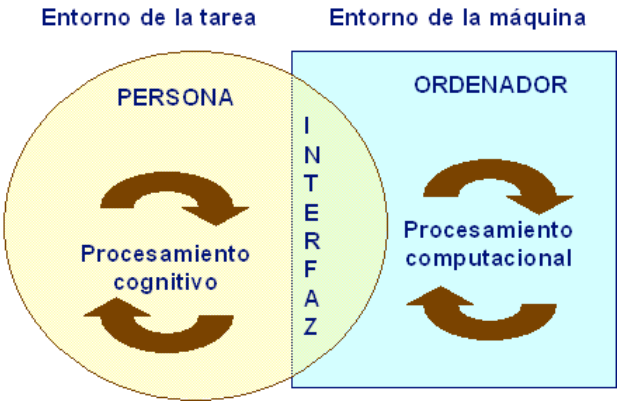


Figura 1.3 Modelo simplificado de la interacción entre el usuario y la máquina, basado en el propuesto en [66]

Tabla 1.1. Ejemplo de proceso de cognitivo del usuario durante la interacción

Tarea: Poner la fecha a una carta en un procesador de textos	
Atención	Observa los botones de la barra de herramientas del procesador de textos tratando de encontrar alguno que inserte la fecha de hoy.
Cognición	Identifica un botón que podría justificar la fecha a la izquierda de la página pero no encuentra uno que inserte la fecha ⇒ tendrá que escribir la fecha y luego justificarla.
Intención	Escribe la fecha, selecciona el texto y pulsa el botón.
Codificación	Observa la respuesta que se produce por parte de la aplicación: la fecha ha sido justificada a la izquierda.

Evaluación	El objetivo ha sido conseguido satisfactoriamente, luego se ha seguido el proceso adecuado.

Como se refleja en la tabla, el usuario realiza una serie de actividades cognitivas (en la columna sombreada de la izquierda) que le hacen llevar a cabo una acción sobre la interfaz (la intención) y, tras recibir la respuesta de la aplicación, realiza otras actividades cognitivas que le hacen llegar a una conclusión sobre el funcionamiento de la aplicación y la consecución de sus objetivos. Dichas conclusiones pueden ser acertadas o no, incluso si el resultado recibido es el esperado, puede que el proceso seguido no sea el óptimo. De hecho en el ejemplo mostrado, el usuario no se ha dado cuenta de que podría existir una opción para insertar la fecha directamente sin tenerla que escribir él, de forma que se eviten errores innecesarios.

El hecho de que el conocimiento de la tarea resida en el usuario y que la máquina deba responder de la manera que éste espera, lleva a la necesidad de contar con dicho usuario durante el proceso de desarrollo. El enfoque de desarrollo que hace que el usuario y sus necesidades reales se conviertan en las directrices del proceso de desarrollo de un producto software se conoce como **diseño centrado en el usuario**.

El diseño centrado en el usuario, término acuñado por Gould y Lewis en 1985, tiene como objetivo maximizar la usabilidad del producto para lo cual se asumen cuatro principios básicos [42]:

- **Focalización temprana y continua en el usuario:** es preciso estudiar las características cognitivas, antropológicas, de actitud y los patrones de comportamiento de los usuarios. Para ello es preciso entender la naturaleza de la tarea que se va a realizar con el producto y los requisitos que ésta impone, por lo que hay que involucrar al usuario en el desarrollo.
- **Medidas empíricas:** los usuarios, ya sean reales o representantes de grupos de usuarios reales, deben enfrentarse a prototipos o maquetas del producto para llevar a cabo las tareas a las que está destinado dicho producto, de tal forma que se puedan recoger y analizar datos válidos sobre la utilidad del sistema.
- **Diseño iterativo:** el modelo de proceso debe permitir iteraciones en las que se tengan en cuenta los datos empíricos recibidos de la interacción del usuario con el producto para mejorarlo.
- **Diseño integrado**: todos los aspectos del diseño de la usabilidad, ya sea la interfaz, la documentación o el plan de implantación, deben evolucionar a la par y no de forma secuencial.

De acuerdo con la norma ISO 13407 (Proceso de Diseño centrado en la Persona para Sistemas Interactivos, *Human-centred design process for interactive systems*), las actividades involucradas en el ciclo de vida del desarrollo de sistemas interactivos son (véase la Figura 1.4): analizar y especificar los contextos de uso o escenarios, especificar los requisitos organizativos así como los de los usuarios, producir soluciones de diseño, y evaluar esas soluciones frente a los requisitos.

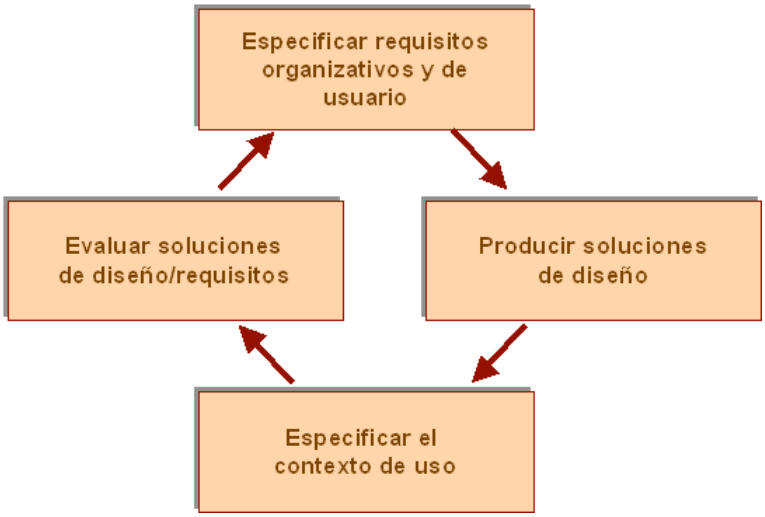


Figura 1.4 Actividades del diseño centrado en el humano y sus relaciones [51]

El modelo de proceso elegido, que determina cómo secuenciar estas actividades dependerá de múltiples factores, pero siempre se deberían emplear métodos que den soporte a las cuatro actividades fundamentales que permiten contar la participación activa del usuario durante todo el proceso de desarrollo para construir un producto de mayor calidad. Algunos aspectos a tener en cuenta durante la planificación de la participación del usuario incluyen:

- la especificación de cuándo y cómo van a participar los usuarios, ya sean estos expertos en el dominio de la aplicación o usuarios reales, debe considerarse desde el comienzo del desarrollo.
- el contexto en el que el usuario realiza su trabajo, y,
- la terminología y la notación a emplear deben ser familiares para el usuario o fáciles de entender y de aprender, de tal manera que éste siempre comprenda lo que se le está mostrando y pueda contrastarlo frente a su conocimiento del entorno de la tarea.

Toda la información proporcionada por los usuarios, ya sea en discusiones sobre el análisis y el diseño o en evaluaciones de prototipos, debe emplearse para mejorar el proceso de interacción convirtiéndose en una valiosa entrada para los procesos de análisis, diseño y construcción.

1.4.2 Equipo de desarrollo pluridisciplinar

El desarrollo de un sistema interactivo es un proceso que tiene múltiples facetas y, además, no todas ellas son de carácter tecnológico. Así, de acuerdo con [64], un sistema interactivo está formado por una serie de niveles, cada uno de los cuales se puede subdividir en otros niveles, tal y como se muestra en la Figura 1.5. En primer lugar, el sistema está formado por el sistema informático y por sus usuarios. Para comprender las necesidades del usuario se pueden requerir conocimientos de

sicología, sociología, antropología e incluso fisiología, especialmente si el desarrollo en cuestión no sólo incluye software sino también hardware. Por otra parte el sistema informático puede dividirse en el software de la aplicación, que será desarrollado por personal fundamentalmente técnico, y la interfaz de usuario, que requerirá de la intervención de especialistas en diseño, ingeniería, lingüística o arte [30].

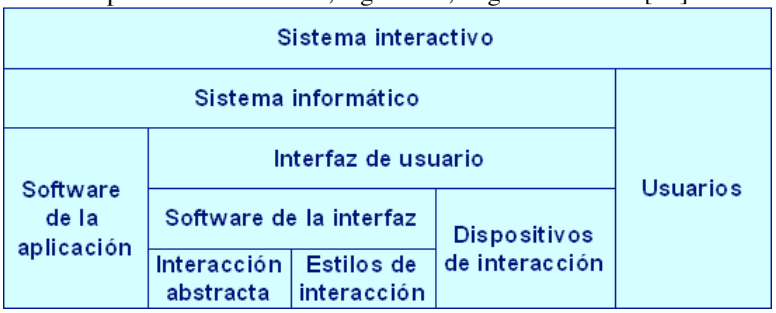


Figura 1.5 Visión modular de un sistema interactivo

Si bien no siempre es preciso ni factible involucrar a tantos tipos de profesionales, si es recomendable que el equipo sea mínimamente pluridisciplinar, contando al menos con:

- técnicos capaces de realizar el análisis, el diseño y la implementación del software de la aplicación,
- especialistas en las necesidades de los usuarios que puedan tomar decisiones sobre las opciones de diseño (psicólogos, sociólogos o, por ejemplo, pedagogos en un sistema destinado a la enseñanza), y
- diseñadores gráficos y otros especialistas en el tratamiento de información multimedia.

En la Tabla 1.2 se muestra las habilidades que deberían cubrirse para el desarrollo de sistemas web, como ejemplo de desarrollo de sistema interactivo. En la tabla se indican el tipo de miembro del equipo de desarrollo, las labores que éste debe realizar y, finalmente, las habilidades y conocimientos requeridos.

Tabla 1.2 Miembros del equipo de desarrollo de aplicaciones web, sus labores y habilidades, ampliadas de [45]

Miembro del equipo	**Labor**	**Habilidades/Conocimientos**
Usuarios	Aportar opiniones y experiencia de uso del producto.	Conocimiento de la tarea a realizar y del dominio en que se lleva a cabo la misma.
Proveedores de contenidos	Producir contenidos multimedia.	Principios de usabilidad, creatividad, diseño gráfico y multimedia, etc.

Ingenieros de la web de nivel 1 (publicación)	Publicar el material en web.	Bases de datos, y lenguajes de programación y marcado para web (HTML, XML, XSL, SMIL, etc.), lenguajes de servidor etc.
Encargados del mantenimiento de la web	Actualizar y mantener los contenidos de información.	Uso de las bases de datos y/o de lenguajes de marcado.
Administrador de la web	Gestionar el servidor web, responsabilizándose de su eficiencia, integridad, seguridad, etc.	Arquitecturas cliente/servidor, comunicaciones, protocolos, mecanismos de seguridad, etc.
Ingenieros de la web de nivel 2 (desarrollo)	Analizar los requisitos, producir la documentación del análisis y del diseño, establecer los procedimientos de operación y de mantenimiento, definir la política de seguridad.	Plataformas hardware/software existentes (características, limitaciones...), técnicas de especificación y modelado, publicación de páginas web, etc.
Ingenieros de la web de nivel 3 (gestión del proyecto)	Gestionar los recursos, humanos y técnicos del proyecto para conseguir que las metas se obtengan en el menor tiempo posible.	Planificación y gestión de proyectos, análisis de riesgos, gestión de calidad, etc.

El hecho de que el equipo de desarrollo sea pluridisciplinar no sólo tiene implicaciones durante los procesos de planificación y gestión del proyecto hipermedia, sino también en el modelo de proceso y el método de desarrollo elegidos, puesto que éstos deberán contar con actividades y productos capaces de fomentar la necesaria cooperación y sinergia entre personas de características, conocimientos y habilidades de diversa índole.

1.4.3 Requisitos de modelado para sistemas hipermedia

En esta sección se analizarán los requisitos que pueden plantearse a un método de desarrollo de aplicaciones hipermedia y web, con objeto de establecer el contexto de este libro así como los aspectos que se van a considerar a la hora de llevar a cabo la evaluación analítica de sus resultados. La sección se inicia con un estudio sobre las características de la hipermedia que deben ser tenidas en cuenta durante el modelado de este tipo de sistemas, recogidas bajo seis perspectivas: navegación, presentación, estructura, comportamiento, acceso y funciones (subsección 1.4.3.1). A continuación se detallarán los requisitos organizados bajo dos epígrafes: de un lado se han extraído una serie de requisitos del campo de la ingeniería del software (subsección 1.4.3.2) y,

de otro, del de las características esenciales y deseables de los sistemas hipermedia (subsección 1.4.3.3).

1.4.3.1 Perspectivas del modelado hipermedia

El modelado de sistemas hipermedia y web involucra varias perspectivas que si bien son complementarias son diferentes y como tal deben ser consideradas de forma independiente en el proceso de desarrollo. En concreto se pueden identificar al menos las seis perspectivas recogidas en la Figura 1.6 [23]. Todas estas vistas deben ser tenidas en cuenta desde distintos niveles de abstracción, de tal forma que se lleve a cabo tanto una autoría de alto nivel más conceptual, es decir lo que habitualmente se conoce como *authoring-in-the-large* [41], así como una autoría de bajo nivel más relacionada con unidades de implementación, es decir el complementario *authoring-in-the-small* [41]. En los siguientes apartados se profundiza en las implicaciones de cada una de estas perspectivas.

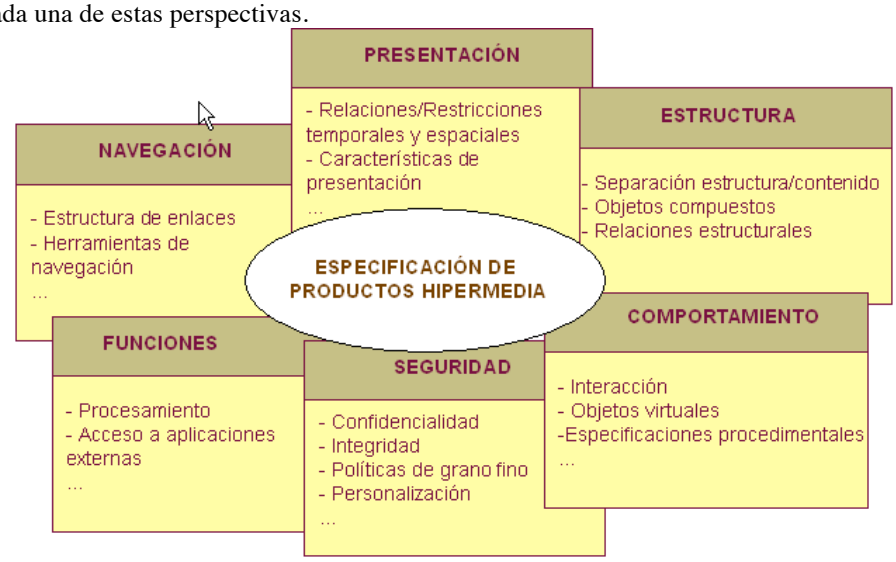


Figura 1.6: Perspectivas para el desarrollo de sistemas hipermedia

Diseño de la navegación

Escribir un hipertexto es superar la estructura secuencial del discurso en favor de otra asociativa por la que el usuario puede navegar a su libre albedrío seleccionando una serie de enlaces o conexiones. Esta es la esencia del hipertexto y, en consecuencia, cualquier método debe ofrecer mecanismos que hagan posible especificar esa red de nodos interconectados que conforma la estructura hipertextual y, de hecho, los primeros modelos de aplicación para hipermedia que aparecieron en los años 90 incorporaban productos específicos para recoger los caminos asociativos y la semántica de la navegación, como por ejemplo HDM [41], las redes de Petri de [78] o el modelo orientado a objetos descrito en [54]. En este punto es importante recalcar que no hay que confundir la estructura hipertextual, es decir, la red asociativa

navegable, con las características estructurales del hipertexto, que vienen dadas por relaciones del tipo parte-todo o generalización-especialización.

Al activar un enlace se puede dar lugar a una gran variedad de resultados, como son: trasladarse a un nuevo tema; mostrar una referencia, una anotación o una definición; presentar una ilustración o esquema; ver un índice, etc. Además, de acuerdo con la forma en que se definan el origen y el destino del enlace, denominados anclas, existen muchos tipos de enlaces, entre los que se cuentan [20]: los enlaces entre nodos y entre posiciones de cualquier tipo de contenido, los enlaces bidireccionales y unidireccionales, los enlaces embebidos, los enlaces virtuales y los enlaces multi-origen o multi-destino (n-arios).

Además, a medida que la estructura asociativa crece se va convirtiendo en un intrincado laberinto en el que el usuario puede llegar a perderse. Para evitar el problema clásico de la desorientación en el hiperespacio se suelen incluir herramientas de ayuda a la navegación, tales como los mapas visuales, los índices activos, las visitas guiadas, las marcas, las huellas o los mecanismos de vuelta atrás [61].

Finalmente, hay que tener en cuenta que el modelado de la navegación no sólo implica la definición declarativa de una serie de enlaces o herramientas, sino que también será preciso incluir especificaciones basadas en eventos o programables con las que modelar elementos que se generan, total o parcialmente, en tiempo de ejecución. Por ejemplo, la estructura espacial que representa una sesión de un usuario concreto, ya sea como lista de nodos o como mapa, requiere una definición procedimental. En este sentido, RMM [49] introdujo las herramientas de navegación condicionales, un concepto que habría que ampliar a los propios enlaces. Además, la semántica de la navegación debe poderse especificar de forma programable para mejorar el soporte a factores tales como las características del usuario, sus acciones, la plataforma hardware o software que se esté empleando, el entorno real de operación o algún tipo de política externa [35]. De hecho la adaptación de la navegación es una técnica básica utilizada en los sistemas hipermedia, ya sea ofreciendo una guía directa, ordenando enlaces o anotándolos [12].

Modelado de la Presentación

La forma en que se presenta la información es sin duda un aspecto básico en un sistema hipermedia, y no sólo por razones estéticas, puesto que la manera y el ritmo de presentación de los contenidos pueden determinar en gran medida el éxito del mismo. Así, hay tener en cuenta que en una composición hipermedia en la que los objetos van apareciendo y desapareciendo del espacio de representación, es preciso establecer una coreografía de los contenidos armónica y eficiente, de manera que se consiga no sólo una presentación estéticamente adecuada sino también cognitivamente eficiente.

Aunque a primera vista podría parecer que la información sobre la presentación sólo puede definirse en las últimas etapas del desarrollo por ser muy concreta, sí que puede indicarse algunas características a nivel conceptual que fuercen a utilizar algunos principios de diseño de la usabilidad de manera que se incite a los diseñadores del software y de los contenidos multimedia a prestar atención a la naturaleza estética de este tipo de aplicaciones [59]. Por ejemplo, se pueden

establecer plantillas que obliguen a mantener una interfaz similar en nodos del mismo tipo o incluir pistas y ayudas visuales.

Los contenidos van a poder ubicarse en diferentes dimensiones o espacios finitos de coordenadas, que serán como mínimo dos: el de la ventana bidimensional (o tridimensional) y el del tiempo. Además, estos contenidos pueden colocarse en un punto concreto de ese espacio (v.g. el vídeo demostración se inicia en el segundo cinco) o bien hacer que su presentación dependa de la presentación de otro u otros elementos (v.g. en cuanto acaba el vídeo demostración se muestra un botón para continuar). Con este fin, es preciso que el método elegido permita establecer relaciones o restricciones espacio-temporales entre contenidos. El uso de ubicaciones relativas frente a absolutas posibilita una mayor independencia con respecto a las características de la plataforma de explotación.

Otro aspecto relevante de cara a la presentación es la posibilidad de separar las características de presentación del contenido en sí, de manera que se potencie la reutilización y la accesibilidad. Cada contenido puede ir acompañado de unas especificaciones de presentación (v.g. tamaño de letra, color, tipografía o incluso estilo retórico para un texto). En esa especificación se puede considerar la posibilidad de recoger la proyección y la modificación de objetos, tal y como se definen en la norma HyTime [19].

Modelado de la estructura

La información de un sistema hipermedia tiene una estructura lógica, distinta de la estructura hipertextual, que está definida por una serie de nodos y de relaciones estructurales establecidas entre dichos nodos. Un nodo es un contenedor abstracto de información [20] en el que se pueden insertar distintos elementos de contenido. Se puede identificar un nodo con una ventana, un marco o una página de un libro electrónico, por ejemplo; mientras que sus contenidos serán los diversos textos, imágenes, fotos, gráficos, vídeos, animaciones, sonidos, etc. que se presenten en los distintos nodos que conforman la aplicación. Esta primera separación estructural, lógica y física, de los contenedores, nodos, y de los contenidos en sí, cada ítem multimedia que se incluye en la aplicación, es útil no sólo porque son elementos conceptualmente distintos, sino porque, además, facilita:

- compartir información por referencia, como por ejemplo se hace en HTML con imágenes y otros elementos multimedia, de manera que se mejore la consistencia, y
- reutilizar las mismas estructuras con distintos contenidos (v.g. una aplicación hipermedia presentada en distintos idiomas tiene la misma estructura pero distintos contenidos).

Pero, además, existe otro nivel de estructuración lógica, pues nodos y contenidos pueden ser simples o compuestos.

Los objetos compuestos estarán formados por otros objetos, simples o compuestos, verificándose entre ellos algún tipo de relación estructural. Si bien la estructura hipertextual es de naturaleza asociativa como se ha discutido con anterioridad, también es cierto que gran parte de los sistemas existentes presentan algún tipo de organización, generalmente jerárquica, que puede modelarse haciendo uso de abstracciones procedentes del modelado de datos tradicionales. De hecho, el modelo

entidad-relación se utiliza en modelos como HDM y métodos como ERMM, el modelado semántico se propone en [70] y técnicas de la orientación a objetos son utilizadas por primera vez en [54]. A este respecto son las abstracciones del modelado semántico, y por ende, del modelado orientado a objetos las que más se acercan a los conceptos propios del domino de la hipermedia puesto que tanto la agregación como la generalización/especialización, por ejemplo, son dos abstracciones presentes en gran parte de los sistemas hipermedia [39].

Modelado del comportamiento

Los sistemas hipermedia se caracterizan por su elevado grado de interactividad, lo cual supone que el sistema debe reaccionar ante determinados eventos. Dichos eventos pueden ser activados tanto por ocurrencias no persistentes (v.g. cuando el usuario selecciona el botón "Parar" se detiene la reproducción de un vídeo) como por estados del sistema o de sus objetos (v.g. siempre que el icono del fichero esté dentro del contorno de la papelera debe desaparecer). Además, dichos eventos pueden ser provocados por acciones realizadas por el usuario (v.g. seleccionar un botón) o por el estado sistema (v.g. tres segundos después de acceder al nodo principal se salta al índice). Así pues, el método de desarrollo debe tener en cuenta la necesidad de especificar estas reacciones como parte esencial del modelado de la aplicación interactiva.

Otro aspecto a tener en cuenta sobre el comportamiento de las aplicaciones interactivas es la necesidad de incluir elementos (nodos, contenidos, enlaces, etc.) que no se indican de manera declarativa sino mediante una especificación procedimental. Es habitual que muchos objetos que se incluyen en el sistema no existan como tales en la base de información del mismo y que no se pueda hacer referencia a ellos por medio de algún tipo de identificador (v.g. un nombre de fichero o un identificador de objeto). Por ejemplo, los resultados de consultas a bases de datos o los enlaces adaptativos cuyo destino depende de lo que haya hecho el usuario previamente (v.g., enlace "Siguiente ejercicio" en un curso adaptativo en el que el siguiente ejercicio propuesto depende de lo que se haya aprendido) necesitan para definirse de algún tipo de procedimiento o de regla que los genere en tiempo de ejecución. En consecuencia, durante el desarrollo también debe tenerse en cuenta este tipo de objetos, normalmente denominados virtuales.

Modelado del acceso

Los sistemas hipermedia, y en especial las aplicaciones basadas en web, han experimentado un considerable crecimiento en los últimos años, y no sólo en el número de sistemas que se han desarrollado sino también en la variedad de los servicios que se ofrecen con objeto de satisfacer las necesidades de todo tipo de usuarios de manera eficiente y efectiva. La portabilidad de los navegadores web, que hace que el acceso a la información se lleve a cabo mediante una interfaz homogénea, ha permitido a muchas compañías y organismos crear sus redes privadas a través de las cuales pueden proporcionar, además de información por la que navegar, servicios especializados a determinados usuarios [5]. Esta situación conlleva la necesidad de dar soporte a políticas de acceso de grano fino, capaces de ofrecer a distintos tipos de

usuarios distintas vistas de la misma información así como distintas habilidades de manipulación en función de sus necesidades y de sus responsabilidades en un contexto concreto [26].

El modelado del acceso se refiere a la especificación de una política de alto nivel [31] en la que se describe quién puede hacer qué en qué contexto y no a los mecanismos físicos que se implementarán para proporcionar un acceso seguro al sistema (v.g. firma digital, cifrado, uso de claves públicas, etc.). Desde esa perspectiva más conceptual que física, el acceso puede analizarse de acuerdo con tres propiedades:

- Confidencialidad: está relacionada con la prevención del acceso no autorizado a la información. El objetivo básico es salvaguardar los datos ante operaciones de lectura por parte de usuarios, ya sean personas o programas, no habilitados para ello [72]. Por ejemplo, "¿quién puede ver las notas de un alumno?" sería una regla de confidencialidad.
- Integridad: supone garantizar que la información no se ha falseado o dañado, esto es, que no se han realizado modificaciones inadecuadas, de forma que cuando el usuario acceda a ella sea completa y exacta [72]. Por ejemplo, "¿quién puede poner las notas de un alumno y en qué plazos?" sería una regla de integridad.
- Disponibilidad: establece que los usuarios habilitados para ello podrán acceder a la información cuando lo requieran [73]. El sistema deberá, además, responder dentro de unos márgenes de tiempo adecuados. Por ejemplo, "cualquier alumno puede acceder a sus notas a través de Internet y desde cualquier dispositivo" sería una regla de disponibilidad.

Confidencialidad e integridad pueden establecerse de acuerdo con una serie de reglas que son computacionalmente tratables, mientras que la disponibilidad involucra parámetros y situaciones que van más allá de los límites del sistema informático (v.g., capacidad de acceso físico a un terminal), por lo que habitualmente no se incluyen como parte del modelado del acceso [10].

El método de desarrollo debe permitir integrar la especificación de la política de acceso, que determine la confidencialidad y la integridad, de tal manera que ésta se exprese en los mismos términos que el resto de las características de la aplicación [26; 31]. Así, si la presentación o la estructura se indican utilizando nodos y contenidos, las reglas de acceso también deben escribirse haciendo uso de esas mismas abstracciones y no de elementos físicos como puedan ser ficheros o directorios en un sistema de ficheros o tablas y atributos de una base de datos. De esta forma los diseñadores podrán decidir aspectos tales como qué enlaces hay que presentar a cada usuario y qué operaciones puede realizar cada uno de ellos con la información del sistema.

En este contexto, uno de los problemas más comunes en los sistemas hipermedia es la complejidad que supone la gestión de los derechos de acceso de manera eficiente, puesto que es preciso manejar autorizaciones para un número ingente de usuarios y de objetos de información y, además, dichas autorizaciones suelen cambiar con bastante frecuencia. Si se opta por un modelo de autorizaciones basado en grupos

y en usuarios individuales, propio de la mayor parte de los sistemas operativos, esta gestión puede llegar a convertirse en una labor inmanejable, trabajosa y tendente a propiciar errores. Frente a esta opción, el control de acceso basado en roles (RBAC, *Role-Based Access Control*) se postula como una alternativa válida para simplificar las tareas de gestión [33], en la que las autorizaciones se definen para roles que representan funciones o responsabilidades propias de la compañía u organización. De hecho, existen estudios empíricos como los recogidos en [37; 53] que demuestran que la implantación de este tipo de políticas de acceso permiten ahorrar tiempo de gestión y, en consecuencia, costes.

Modelado de funciones

El funcionamiento de la aplicación debe también especificarse como en cualquier otro tipo de producto software. En este caso dentro del funcionamiento no se considera la navegación, que está contemplada aparte por ser una función esencial en los sistemas hipermedia y web, sino todos aquellos procesos no relacionados con la navegación y que cada vez son más frecuentes en los sistemas hipermedia y web (v.g., acceso a sistemas externos, validación de datos, código embebido en los nodos, etc.). Por ejemplo, los *applets* normalmente incluidos en las páginas web deben ser diseñados, para lo que cualquier técnica de modelado de procesos sería de utilidad.

1.4.3.2 Requisitos propios de la ingeniería del software

Una vez analizadas las principales perspectivas del modelado de hipermedia, se va a pasar a establecer algunos requisitos que deberían exigirse a un método de desarrollo. En primer lugar, como método en sí existen una serie de características que debe tener por el hecho de tratarse de una herramienta para llevar a cabo un proceso propio de la ingeniería del software y que se detallan en esta sección. Además, por estar destinado al desarrollo de sistemas hipermedia también existen otros requisitos específicos de ese dominio que se enumeran en la sección 1.4.3.3.

De acuerdo con [3; 28; 77] cualquier método de diseño de software debe:

- Describir un proceso formal que guíe el desarrollo de la aplicación software.
- Asumir un modelo de referencia que describa el mundo real y lo traslade a entidades físicas.
- Proporcionar productos para especificar los requisitos del sistema.
- Incluir reglas de validación para cada producto.
- Mantener relaciones de integridad entre productos.
- Facilitar la reutilización del diseño.
- Contar con herramientas software de soporte al desarrollo

A continuación se describen brevemente cada uno de estos requisitos.

Describir un proceso formal que guíe el desarrollo de la aplicación software

Esto implica que el método debería describir las fases o etapas por las que atraviesa el desarrollo, cómo estas fases se integran en el modelo de proceso y qué productos o artefactos se crean en cada momento. En cuanto a los modelos de proceso, como ya se ha dicho en la sección 1.3, aquellos basados en el uso de prototipos, ya sean

iterativos, progresivos o evolutivos [27] o el modelo en estrella [64], son los más adecuados puesto que permiten ir evaluando las soluciones propuestas sin necesidad de haber finalizado la implementación del sistema e ir detectando requisitos durante el propio desarrollo.

Asumir un modelo que describa el mundo real y lo traslade a entidades físicas
Un modelo es una forma de abstracción que permite representar la esencia del diseño. Existen modelos como el entidad-relación o los diagramas y elementos de UML que pueden emplearse para sistemas hipermedia, pero estos modelos no proporcionan elementos propios de la hipermedia como son el nodo, el enlace, el ancla, los contenidos o las relaciones espacio-temporales. A tal efecto existen modelos de referencia para hipermedia, como por ejemplo Dexter [43], Amsterdam [46] o Labyrinth [22, 24], y modelos de aplicación como HDM [41], HDM-lite [34] o las redes de petri para hipermedia de [35; 78].

Proporcionar productos para especificar los requisitos del sistema
Estos requisitos se han clasificado tradicionalmente en funcionales y no funcionales [48]. Mientras que los primeros establecen funciones que el sistema o producto debe realizar, los segundos imponen restricciones a los requisitos funcionales relacionadas con la eficiencia, consistencia, reusabilidad, flexibilidad, adecuación a estándares, etc. En el caso de los sistemas interactivos se incluye un tercer tipo, los requisitos de *usabilidad* que determinan qué se va a entender por un nivel aceptable de utilización y de aceptación del producto final por parte del usuario [63] y pueden hacer referencia a aspectos tales como la estética, legibilidad, consistencia, auto-evidencia o predecibilidad [40].

Incluir reglas de validación para cada producto
El método debe considerar cómo se va a establecer si cada uno de los productos que se crean para modelar un sistema concreto son conceptualmente correctos y completos, por lo que debe proporcionar reglas de validación. Por ejemplo, si un nodo puede componerse de otros utilizando diferentes abstracciones hay que establecer si se permiten o no ciclos, es decir, que un nodo se componga a sí mismo directa o indirectamente.

Mantener relaciones de integridad entre productos
Si se pretende ofrecer un entorno integrado en el que especificar todos los requisitos hay que considerar que estos rara vez son autosuficientes, en el sentido de que pueden ser resueltos utilizando un único producto. En consecuencia, el método debe ofrecer mecanismos para poder trazar los requisitos y acceder a todos los productos que están relacionados con ellos, así como reglas de validación entre diferentes productos que permitan garantizar la consistencia y completitud del diseño [58]. Por ejemplo, una regla de este tipo podría analizar si cada nodo que forma parte de la estructura lógica del sistema tiene una especificación de sus componentes (contenidos o nodos) en la que se incluyan, además, los enlaces que aparecen en el producto en el que se recoge la estructura hipertextual.

Facilitar la reutilización del diseño
El diseño es un proceso creativo que más que de un libro se aprende desde la experiencia de otros diseñadores. En este sentido, los patrones de diseño [38] describen problemas recurrentes y proporcionan el núcleo de una solución eficiente y elegante que puede ser empleada tantas veces como sea necesario en diferentes contextos. Los patrones para hipermedia son abordados en los capítulos 10, 11 y 12. Otra opción es la reutilización de componentes de diseño previamente elaborados como en [36].

Contar con herramientas software de soporte al desarrollo
El desarrollo de un sistema interactivo es un proceso que tiene múltiples facetas y, además, no todas ellas son de carácter tecnológico. En consecuencia, el equipo de desarrollo estará formado por personas con distinta formación que deben interactuar, entre los que se incluyen diseñadores multimedia, ingenieros y programadores, usuarios y especialistas en las necesidades de los usuarios que puedan tomar decisiones sobre las opciones de diseño (psicólogos, sociólogos o, por ejemplo, pedagogos en un sistema destinado a la enseñanza). Para facilitar la comunicación entre miembros del equipo con habilidades y conocimiento tan disperso es preciso que el método ofrezca productos con una notación sencilla y que, a ser posible, esté complementado con un entorno software automatizado en el que cada miembro pueda realizar su trabajo. Esta entorno o herramienta CASE debe permitir generar y validar los productos propios del método, generar documentación sobre el sistema y producir de forma automática o semi-automática prototipos [58].

1.4.3.3 Requisitos relacionados con la tecnología hipermedia
Además los requisitos expuestos en la sección anterior hay otros que se derivan del campo de la ingeniería hipermedia, entre los que se cuentan [25; 55; 59; 67] los siguientes:

- Permitir especificar el dominio del problema en términos de componentes hipermedia
- Dar soporte al diseño ascendente y descendente
- Hacer posible la evaluación del sistema
- Ofrecer herramientas para modelar todas las características de un sistema hipermedia

Permitir especificar el dominio del problema en términos de componentes hipermedia
Al modelar el sistema es preciso contar con elementos propios del dominio de aplicación tales como nodos, contenidos, enlaces y anclas. Un nodo es un contenedor abstracto al que se pueden asociar diferentes fragmentos de información multimedia denominados contenidos [79]. Un enlace es una conexión navegable entre dos nodos o contenidos que se define entre dos conjuntos de anclas [43]. Las anclas se enclavan en áreas de nodos o contenidos y delimitan el origen o el destino de un enlace.

Estos cuatro elementos deben ser considerados como entidades de diseño independientes, con objeto de mantener de forma separada la información, la

estructura lógica, la navegación y la presentación. Esta separación facilitará la reutilización de componentes y, además, ayuda a mantener la consistencia del sistema y hace posible que se puedan ofrecer distintas vistas del mismo nodo a distintos usuarios [16]. Esto supone que los modelos subyacentes a los métodos de desarrollo deberían incluir al menos estos componentes y, además, ofrecer una descripción detallada de qué se entiende por cada uno de ellos y de su traslación a entidades de implementación.

Dar soporte al diseño ascendente y descendente
Tal y como se argumenta en [59] un buen método de diseño para hipermedia debe tener en cuenta que el proceso de desarrollo de este tipo de aplicaciones es bastante flexible, tanto por la naturaleza de las mismas como por las características del equipo de trabajo que, como se ha comentado con anterioridad, tiene una composición multidisciplinar. Por ello, es preciso que permita hacer un diseño ascendente, que permita partir de prototipos para abstraer un modelo conceptual, así como descendente, es decir, que haga posible el proceso inverso. Esta libertad de enfoque incrementará notablemente la utilidad del método, puesto que se adaptará a las necesidades de diferentes proyectos de desarrollo.

Hacer posible la evaluación del sistema
Los sistemas hipermedia se caracterizan por la complejidad de sus interfaces y por los aspectos estéticos y cognitivos de sus contenidos [59]. Consecuentemente, se requiere un proceso de desarrollo centrado en el usuario en el que la evaluación sea una fase fundamental. La evaluación tiene como misión primordial asegurar que las aplicaciones se han diseñado teniendo en cuenta las necesidades de sus usuarios finales, y no sólo las de los desarrolladores. Este proceso va a proporcionar información que permita comprobar si los mecanismos de interacción se han diseñado correctamente, detectando aquellas deficiencias que haya que solventar o proponiendo mejoras [6].

Ofrecer herramientas para modelar todas las características de un sistema hipermedia
El método debe contar con productos, ya sean diagramas u otros artefactos de especificación, con los que se puedan recoger todas las características de los sistemas hipermedia, ya estén éstas relacionadas con la forma en que debe presentarse la información, con la estructura lógica del sistema, con las posibilidades de interacción ofrecidas al usuario, con las formas de acceso ofrecidas y las restricciones vigentes a tal efecto, con los procesos a que se da soporte o con los caminos y herramientas de navegación incluidos en el sistema.

1.5 Métodos de desarrollo de sistemas hipermedia y web

En esta sección se describen brevemente algunos métodos para hipermedia y web y que no están incorporados en este libro pero que son relevantes en el área de la

Ingeniería de la Web. En concreto, se revisarán ERMM, WSDM, WebML y UWE. Ni HDM ni su última versión, HDM-lite, se han incluido en este estudio puesto que ambos modelos definen una serie de primitivas para especificar la estructura y el comportamiento de los sistemas pero no un proceso de desarrollo, por lo cual no pueden considerarse métodos sino más bien modelos de aplicación. Un método ofrece un marco racional que guía al diseñador desde su concepción inicial del problema hacia una solución lógica y formal para lo cual:

- Debe especificar los pasos a seguir y las actividades a realizar para desarrollar la aplicación
- Debe proponer una serie de herramientas o modelos con los que especificar todas las características de la aplicación

Otros métodos como Ariadne, OO-H y OOHDM se revisarán en detalle en los capítulos 2, 3 y 11, respectivamente.

1.5.1 Extended Relationship Management Methodology (ERMM)

El método RMM fué propuesto por primera vez en [49], si bien se trataba ésta de una versión con múltiples limitaciones que al ser detectadas dieron lugar a una versión extendida, ERMM presentada en [50]. Se trata, probablemente, del único método para hipermedia que parece cubrir todo el ciclo de desarrollo, desde el estudio de factibilidad hasta la evaluación del sistema, aunque sólo propone actividades y productos concretos para las fases de análisis y de diseño.

El análisis se realiza por medio de un diagrama entidad-relación en el que sólo se permiten relaciones con cardinalidades 1→1 ó 1→N y tampoco es posible establecer relaciones reflexivas.

Con respecto al diseño, una de las características más relevantes de este método es que éste se hace tanto de forma ascendente como descendente, ofreciendo una interesante forma de realizar una verificación. El diseño descendente empieza con la construcción de un Diagrama de Aplicación descendente, que es un esquema de las unidades de presentación (equiparables a ventanas) y de los enlaces que existen entre las mismas. A continuación se compone cada una de esas unidades partiendo de las entidades del diagrama E-R, generando los denominados *m-slices*. En los *m-slices* se especifican los contenidos, enlaces, herramientas de navegación y funciones asociadas a cada unidad. El diseño ascendente toma como punto de partida los *m-slices* y genera un nuevo Diagrama de Aplicación ascendente que contrasta con el descendente.

Para el resto de las fases, estudio de factibilidad, implementación, pruebas y evaluación no se propone ningún tipo de guías.

1.5.2 Web Site Design Method (WSDM)

WSDM [18] es un método centrado en el usuario más que en los datos y que está destinado a sistemas web, por lo que algunas características de los sistemas hipermedia, como la sincronización de contenidos, no son tenidas en cuenta. Este

método está fundamentalmente orientado a dar soporte al desarrollo de sistemas personalizables, haciendo del modelado de los usuarios una fase esencial.

Para el desarrollo de sistemas web se proponen las fases de Modelado de Usuarios, Diseño Conceptual, Diseño de la Implementación e Implementación. En la primera fase se analizan y clasifican los tipos de usuarios en clases, atendiendo a los requisitos que éstos tienen en cuanto al uso del sistema se refiere. Cada clase de usuarios puede dividirse en diferentes perspectivas que respondan a distintos requisitos de usabilidad, de tal forma que el diseño se hará en función de esas clases y perspectivas. El Diseño Conceptual tiene como misión modelar los requisitos de información de cada clase de usuarios, generándose un Modelo de Objetos de Usuario para cada una de ellas junto con un Modelo de Navegación para cada perspectiva de cada clase. El Diseño de la Implementación consiste en modelar la apariencia de la aplicación que finalmente deberá implementarse haciendo uso de tecnologías propias de los entornos web.

1.5.3 Web Modeling Language (WebML)

El proceso propuesto en WebML [16] está orientado a diseñar sitios web desde una perspectiva de alto nivel y sin entrar en detalles sobre la arquitectura de los mismos. Se trata de un diseño iterativo que pretende guiar al desarrollador desde del proceso de recogida de requisitos hasta el diseño personalizado de la aplicación. Cada ciclo de diseño se compone de las siguientes actividades: Recogida de Requisitos, Diseño de Datos, Diseño Abstracto del Hipertexto (respondiendo al concepto de *in the large* introducido en HDM), Diseño Detallado del Hipertexto (atendiendo al concepto de *in the small* de HDM), Diseño de la Presentación, Diseño de los Usuarios y de los Grupos y Personalización del Diseño. A tal objeto propone cuatro herramientas: el Modelo Estructural, el Modelo de Hipertexto, el Modelo de Presentación y el Modelo de Personalización.

El Modelo Estructural representa los datos que ofrecerá el sitio web así como las relaciones estructurales que existen entre ellos, para lo cual se aconseja emplear cualquier técnica al efecto, como por ejemplo el modelo entidad-relación [17] o los diagramas de clases de UML [9]. Dentro de este modelo se incluyen los usuarios y los grupos como un tipo de entidad predefinida que servirá para personalizar el sitio web. El Modelo de Hipertexto describe qué páginas y contenidos componen el sitio y cómo están enlazadas a través de dos submodelos: el Modelo de Composición y el Modelo de Navegación. El Modelo de Presentación es empleado para especificar la apariencia de las páginas de una forma independiente de las tecnologías de presentación y de la plataforma de implementación, por medio de una sintaxis abstracta basada en XML. Finalmente, el Modelo de Personalización está basado en el esquema estructural en el que se han definido los tipos de usuarios y al que se pueden añadir reglas utilizando una especie de lenguaje OQL (*Object Query Language*) que indicarán como derivar el contenido atendiendo a ciertas características del usuario.

1.5.4 UML-based Web Engineering methodology (UWE)

Se trata de un método que hace uso de técnicas procedentes de la orientación a objetos para especificar aplicaciones hipermedia. UWE [47] plantea una aproximación iterativa y progresiva cuyas actividades fundamentales son el análisis de requisitos y el diseño conceptual, de la navegación y de la presentación.
Los elementos hipermedia se representan por medio de elementos propios de los Diagramas de clases UML. Así por ejemplo, los nodos son clases, los enlaces son asociaciones estereotipadas y las ayudas a la navegación (tales como índices o mapas) son clases estereotipadas. Para modelar aspectos dinámicos se hace uso de modelos de tarea y diagramas de estado, mientras la navegación y la presentación se representan por medio de UML y de estereotipos creados al efecto.

1.6 El libro "Ingeniería de la web y patrones de diseño"

Este libro nace con el objetivo de dar una panorámica global sobre el proceso de construcción de sistemas y aplicaciones hipermedia y, en particular, de los orientados a la web, haciendo especial hincapié en aspectos relevantes como el diseño, la reutilización de conocimiento, la calidad y la seguridad. Además, se pretende que sea un libro académico y, al mismo tiempo, práctico por lo que cada uno de los capítulos incluye distintas experiencias de aplicación y gran número de ejemplos destinados a conseguir que el lector comprenda los beneficios de cada una de las aproximaciones presentadas.

Los capítulos 2, 3 y 4 presentan tres alternativas distintas de diseño y construcción de sistemas web, el modelado a través de mecanismos de la tecnología hipermedia mediante Ariadne, el uso de la orientación a objetos mediante OO-H, y la perspectiva de desarrollo basada en documentos. El capítulo 5 proporciona una visión distinta de las tres propuestas anteriores denominada MIDAS, en la que el eje central es una arquitectura dirigida por modelos basada en MDA (*Model Driven Architecture*) y un proceso de desarrollo basado en el uso de técnicas procedentes de metodologías ágiles. Un valor añadido en el desarrollo de muchos sistemas web es la posibilidad de describir entornos virtuales que ayuden al usuario a interactuar con el sistema, por eso en el capítulo 6 se presenta una alternativa, denominada SENDA, para el desarrollo de este tipo de sistemas, así como las limitaciones que conllevan el desarrollo de este tipo de entornos en la actualidad.

Los tres capítulos siguientes se centran en tres aspectos fundamentales en el desarrollo de sistemas orientados a la web como son la seguridad, la calidad y la accesibilidad. En el capítulo 7 se hace especial hincapié en la necesidad de integrar el diseño de la seguridad del sistema web en el diseño del propio sistema empleando los modelos adecuados que permitan reflejar la política de seguridad de la organización. La calidad del sistema es un punto clave del desarrollo de cualquier sistema web, por lo que en el capítulo 8 se trata este tema en profundidad, centrándose en la calidad desde el punto de vista de la usabilidad del sistema y cómo se puede integrar su evaluación durante el proceso de desarrollo. También con el objetivo de ser integrado durante todo el proceso de desarrollo, en el capítulo 9 se centra en el concepto de

accesibilidad y cómo se mejora la usabilidad del sistema si éste ha sido desarrollado de manera que todo tipo de público puede acceder a él.

Los tres últimos capítulos presentan los patrones como una herramienta que puede facilitar el desarrollo de sistemas interactivos y sistemas web, que son soluciones que se han encontrado adecuadas a problemas que se producen recurrentemente. En el capítulo 10, se discutirán los conceptos básicos de los patrones para pasar a discutir los diferentes usos en el campo de la interfaz de usuario. Los patrones en el campo de la Ingeniería de la Web se introducen en el capítulo 11, en el que los autores presentan algunos ejemplos de patrones y cómo pueden ser utilizados durante la construcción de aplicaciones web. Por último, el capítulo 12 presenta una serie de mecanismos dirigidos a agilizar la recuperación y aplicación de los patrones en conjunción con los métodos de diseño.

1.8 Referencias

1. Aedo, I, Díaz, P., Sicilia, M.A., Colmenar, A., Losada, P., Mur, F., Castro, M. y Peire, J. (2004): Sistemas multimedia: análisis, diseño y evaluación. Editorial UNED
2. Anderson, K. M.,Taylor, R. N. and Whitehead, E. J. Chimera: Hypertext for Heterogeneous Software Environments, ECHT '94 Proceedings. 94-107.
3. Avison, D.E. y Fitzgerald, G. Information systems development: techniques and tools. Blackwell Scientific Publications.
4. Baresi, L., Morasca, S. y Paolini, P. An empirical study on the Design Effort of Web Applications. Proceedings of 3rd International Conference on Web Information Systems Engineering (WISE'02). IEEE Computer Society.
5. Barkley, J., Cincotta , A., Ferraiolo, D., Gavrilla, S. y Khun, R. Role based access control for the world wide web. Proccedings of 20th National Computer Security Conference. 331-340.
6. Benyon, D., Davies, G., Keller, L. y Rogers, Y. A Guide to Usability- Usability Now!". Milton Keynes: The Open University. Gran Bretaña.
7. Bernstein, M. Patterns of Hypertext. Proc. of Hypertext'98. Pittsburgh. 21-29.
8. D. M. Berry. Academic Legitimacy of the Software Engineering Discipline, Technical Report CMU/SEI-92-TR-34, Software Engineering Institute, Carnegie Mellon University, Pittsburgh (Pennsylvania, EE.UU.), Noviembre 1992.
9. Booch, G., Jacobson, I. y Rumbaugh, J. The Unified Modeling Language. Addison-Wesley.
10. Brinkley, D.L. y Schell, R.R.: Concepts and Terminology for Computer Security. In "Information Security. A collection of essays" Ed. Abrams. M.D., Jajodia, S. and Podell, H.J. IEEE Computer Society Press.40-97.
11. Brusilovsky, P., Kobsa, A. y Vassileva, J. Adaptive Hypertext and Hypermedia. Kluwer Academic Publishers.
12. Brusilovsky, P. Methods and techniques of adaptive hypermedia. User Modeling and User Adapted Interaction, 6(2-3). 87-129.
13. Bush, V. As we may think. Atlantic Monthly. 176 (1-Julio).101-108.

14. Campbell, B. y Goodman, J. M.: HAM: A general purpose hypertext abstract machine' Communications of the ACM 31 (7). 856-861.
15. Ceri, S., Fraternali, P. y Paraboschi, S. Design principles for data-intensive web sites. SIGMOD Record, 28(1). 84–89.
16. Ceri, S., Fraternali, P. and Bongio, A. Web modeling language (WebML): a modeling language for designing web sites. WWW9 / Computer Networks, 33(1-6). 137–157.
17. Chen, P. The entity-relationship model – toward a unified view of data. ACM Trans. on Database Systems. 9–36.
18. De Troyer, O. y Leune, C. WSDM: a user centered design method for web sites. Computer Networks and ISDN Systems, 30(1–7). 85–94.
19. DeRose, S. y Durand, D. Making Hypermedia Work: A User's Guide to HyTime. Kluwer Academic Publishers.
20. Díaz, P., Catenazzi, N. y Aedo, I. De la multimedia a la hipermedia. Ed. Rama, Madrid. 1996.
21. P. Díaz, I. Aedo y S. Montero. Ariadne, a development method for hypermedia. Dexa 2001. LNCS 2113, pp. 764-774.
22. Díaz, P., Aedo I. and Panetsos, F. Labyrinth, an abstract model for hypermedia applications. description of its static components. Information Systems, 22(8). 447–464.
23. Díaz,, P., Aedo, I. and Panetsos, F. A methodological framework for the conceptual design of hypermedia systems. In Proc. of the fifth conference on Hypertexts and hypermedia: products, tools and methods (H2PTM 99). 213–228.
24. Díaz P., Aedo, I. and Panetsos, F. Modelling the dynamic behaviour of hypermedia applications. IEEE Transactions on Software Engineering, 27(6). 550–572.
25. Díaz, P., Aedo, I. y Montero, S. Ariadne, a development method for hypermedia. In proceedings of Dexa 2001, volume 2113 of Lecture Notes in Computer Science, pages 764–774, 2001.
26. Díaz,, P., Aedo, I. and Panetsos, F. Modelling security policies in hypermedia and web-based applications. In Web Engineering: Managing diversity and complexity of web application development, volume 2016 of LNCS. Murugesan, S. and Deshpande, Y. Eds. Springer-Verlag. 90–104.
27. Dix, A., Finlay, J., Abowd, G. y Beale, R. Human-Computer Interaction, 2nd Ed. Prentice Hall. 1998.
28. Duffy, T.M., Palmer, J.E. y Mehlenbacher, B. On line help: design and evaluation. Norwood, N.J.
29. Engelbart, D. C. A Conceptual Framework for the Augmentarion of Man's Intellect. Ed. Greif, I. "Computer-Supported Cooperative Work: A Book of Readings". Morgan Kaufmann Publishers Inc. 35-66.
30. C. Faulkner. The essence of human-computer interaction. Prentice Hall, 1998.
31. Fernández, E. B., Krishnakumar, R.N., Larrondo-Petrie, M.M. y Xu, Y.: High-level Security Issues in Multimedia/Hypertext Systems. Communications and Multimedia Security II. P. Horster (ed.), Chapman & Hall. 13-24.

32. Ferraiolo, D.F., Barkley, J.F. and Kuhn, D.R. A role-based access control model and reference implementation within a corporate intranet. ACM Trans. On Information and Systems Security, 2(1). 34–64.
33. Ferraiolo, D. y Kuhn, R.: Role-Based Access Control. Proc. of the 15th National Computer Security Conference. 554-563.
34. Fraternali, P. y Paolini, P. Model-driven development of web applications: the autoweb system. ACM Transactions on Office Information Systems, 18(4). 323–282.
35. Furuta, R. y Na, J. Applying programmable browsing semantics within the context of the world-wide web. Proceedings of Hypertext'02, University of Maryland, June 11th-15th. ACM Press. 23–24.
36. Gaedke, M. y Gräf, G. Development and evolution of web-applications using the WebComposition process model. In International Workshop on Web Engineering at the 9th International World-Wide Web Conference (WWW9). LNCS 2016. 58–76.
37. Gallaher, M.P., O'Connor, A.C. y Kropp, B. The economic impact of Role-based Access Control. Planning Report 02-1. National Institute of Standards & Technology. Marzo.
38. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley.
39. Garg, P. K. Abstraction mechanisms in hypertext. Communications of the ACM. 31 (7). 862-870.
40. Garzotto, F., Mainetti L. y Paolini, P. Hypermedia design, analysis, and evaluation issues. Communications of the ACM, 38(8). 74–86.
41. Garzotto, F., Paolini, P. and Schwbe, D. HDM- a model-based approach to hypertext application design. ACM Transactions on Information Systems, 11(1).1–26.
42. J. D. Gould, S. J. Boies y J. P. Ukelson. How to design usable systems. En Handbook of Human Computer Interaction, pp 231-254. Elsevier Science, 1997.
43. Halasz, F. G. y Schwartz, M.: The Dexter Hypertext Reference Model. Proc. of World Conference of Hypertext 1990. 95-133.
44. Hall , W., Davis, H. y Hutchings, G. Rethinking Hypermedia: The Microcosm Approach, Kluwer Academic Publishers.
45. S. Hansen, Y. Deshpande y S. Murugesan, A Skills Hierarchy for Web Information System Development. Web Engineering: Managing Diversity and Complexity of Web Application Development, Eds. San Murugesan y YogeshDeshpande, LNCS, 2016, Springer Verlag, 2001, pp 223 -236.
46. Hardman, L. Bulterman, D. y Van Rossum, G. The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. Communications of the ACM, 37(2). 50-62.
47. Hennicker, R. y Koch, N. A UML-based methodology for hypermedia design. Proc. UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York (Reino Unido), Octubre. Eds. Andy Evans, Stuart Kent y Bran Selic. LCNS, vol. 1939. Springer Verlag. 410–424.
48. IEEE Standard Glossary of Software Enginnering Terminology. IEEE Std. 610.12-1990.

49. Isakowitz, T., Stohr, E. A. and Balasubramanian, P. RMM: a methodology for structured hypermedia design. Comm. of the ACM, 38(8). 34–44.
50. Isakowitz, T. and Kamis, A. and Koufaris, M. The extended RMM methodology for web publishing. Technical Report Working Paper IS98 -18, Center for Research on Information Systems.
51. ISO 13407: Human-centred design processes for interactive systems
52. ISO Technical Committee JTC 1/SC 29: Coding of audio, picture, multimedia and hypermedia information. Disponible en http://www.iso.ch/
53. Kropp, B. y Gallaher, M. P. Access to cost savings: Role-based access control systems can save organizations time and money. Information Security Maganize (Abril). http://www.infosecuritymag.com/articles/april01/cover.shtml#case_study
54. Lange, D. B. An object-oriented design method for hypermedia information systems. Proceedings of the 27th Annual Hawaii International Conference on System Sciences. 366–375.
55. Lowe, D. y Hall, W. Hypermedia and the web: an engineering approach. John Wiley & Sons. 1999.
56. Lowe, D. Y Webby, R. The impact process modelling project work in progress. 1st International Workshop on Hypermedia Development. Hypertext'98, Pittsburgh, PA, USA, June 20-24.
57. Mayhew, D.J. The usability engineeering lifecycle : a practitioner's handbook for user interface design. Morgan Kaufmann .
58. Montero, S., Díaz, P. y Aedo, I. A framework for the analysis and comparison of hypermedia design methods. Proc. of The IASTED International Conference on Software Engineering (SE'2003).1053–1058.
59. Nanard, J. y Nanard, M. Hypertext design environments and the hypertext design process. Comm. of the ACM, 38(8).49–56.
60. Nielsen, J. Usability Engineering. Academic Press. EE.UU, 1993.
61. Nielsen, J. Hypertext and Hypermedia: the Internet and Beyond. Academic Press.
62. Nürnberg, P. J. y Leggett, J. J. A Vision for Open Hypermedia Systems. Journal of Digital information, 1(2).
63. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. y Carey, T.. Human Computer Interaction. Addison-Wesley. 1994.
64. Preece, J., Rogers, Y. y Sharp, H. Interaction Design: beyond human computer interaction. John Wiley &Sons. 2002.
65. Rada, R. Hypertext: from Text to Expertext, McGraw-Hill.
66. R. Rada. Interactive Media. Springer-Verlag, 1995.
67. Retschitzegger,W. y Schwinger,W. Towards modeling of dataweb applications - a requirements' perspective. In Proc. of the Americas Conferenc on Information Systems (AMCIS) Long Beach California, vol.1.
68. Rossi, G., Schwabe, D. y Lyardet, F. Integrating patterns into the hypermedia development process. New Review of Hypermedia and Multimedia, 5. 59–80.
69. Rumbaugh, J., Blaha, M., Premerlani, W. , Eddy, F. and Lorensen, W. Object Oriented Modeling and Design. Prentice Hall Inc.
70. Schnase, J.L., Leggett, J.J., Hicks, D.L. y Szabo, R.L. Semantic data modelling of hypermedia associations. ACM TOIS, 11(1). 27–50.

71. Schwabe, D. y Rossi, G. An object oriented approach to web-based application design. *Theory and practice of object system*s, 4(4):207–225.
72. Shandu, R.S. y Jajodia, S. Integrity principles and mechanisms in database management systems. Computer and Security, 10. 413–427.
73. Shandu, R.S. Lattice-based access control models. IEEE Computer, 10. 9–19.
74. Shipman, F., Moore, J.M., Maloor, P., Hsieh, H. and Akkapeddi, R. Semantics happen: knowledge building is spatial hypertext. In Proceedings The Thirteenth ACM Conference on Hypertext and Hypermedia, Maryland, USA.,25-34.
75. Sinclair, P., S., Martinez, K., Millard, D. y Weal, M. J. Links in the Palm of your Hand: Tangible Hypermedia using Augmented Reality. In Proceedings The Thirteenth ACM Conference on Hypertext and Hypermedia, Maryland, USA., 127-136,
76. Slowinski, M., Kennedy, T. and Kennedy, T. SMIL: Adding Multimedia to the Web. Sams Publishing.
77. Sommerville, I. y Sawyer, P. Requirements engineering: a good practice guide. Wiley.
78. Stotts, P.D. y Furuta, R. Petri-net based hypertext: Document structure with browsing semantics. ACM TOIS, 7(1). 3–29.
79. Tompa, F. A Data Model for Flexible Hypertext Database Systems. ACM Transactions on Information Systems, 7(1). 85-100.
80. Van Duyne, D.K, Landay, J.A., Hong, J.I (2002): The design of sites. Addison-Wesley.
81. Wiil, U.K,, Bouvin, N.O., Larsen, D., De Roure, D.C. and Thompson, M.K. Peer-to-Peer Hypertext. Departmental Record 6, Electronics and Computer Science Dept., University of Southampton.
82. Zellweger, P.T. ,Newman, P. y Mangen, A. Reading and Writing Fluid Hypertext Narratives. . In Proceedings The Thirteenth ACM Conference on Hypertext and Hypermedia, Maryland, USA.,45-54.

2 Diseño de hipermedia y web con ADM

Paloma Díaz, Ignacio Aedo y Susana Montero

Laboratorio DEI.Departamento de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad 30, 28911 Leganés
aedo@ia.uc3m.es, pdp@inf.uc3m.es, smontero@inf.uc3m.es
http://dei.inf.uc3m.es

2.1 Introducción

Como ya se ha discutido en el primer capítulo de este libro, es evidente que el desarrollo a gran escala de sistemas hipermedia y web debe llevarse a cabo siguiendo un proceso sistemático y bien definido [7] y no de forma artesanal. En este sentido, la experiencia en ingeniería de software es útil, fundamentalmente por la disciplina que impone al desarrollo y las capacidades de abstracción y modelado que permite adquirir, pero se requieren métodos que estén fundamentados en elementos y conceptos propios del dominio de la hipermedia, como puedan ser el nodo, el enlace o las relaciones temporales entre contenidos, y que recojan la experiencia de otros autores en el desarrollo de este tipo de sistemas. De hecho, en los últimos años han proliferado métodos para hipermedia o web, algunos de los cuáles se revisan en este libro. En concreto, este capítulo presenta un método denominado *Ariadne Development Method* (ADM), que plantea un proceso sistemático, integrador e independiente de la plataforma de implementación para modelar y evaluar aplicaciones y sistemas hipermedia. ADM aborda aspectos clave como son:

- Contemplar los elementos de contenido multimedia de forma rigurosa, entendiendo que su inclusión no sólo implica disponer de aplicaciones software que permitan su reproducción, sino también tener en cuenta aspectos estéticos y de *usabilidad*, como por ejemplo la necesidad de establecer restricciones espaciales y temporales entre ellos, de manera que se pueda producir una presentación armónica y eficiente.
- Integrar la especificación de requisitos funcionales no relacionados con las facilidades de navegación.
- Incorporar a los usuarios del sistema en el diseño resultante, de manera que este modelado facilite el modelado de sistemas adaptativos o de reglas de acceso a la información.
- Aportar medios para establecer políticas de acceso con las que se pueda determinar quién puede hacer qué operaciones sobre determinados objetos del sistema.

El resto del capítulo se inicia con un análisis de las principales características de ADM en la sección 2.2. A continuación, la sección 2.3 recoge una descripción detallada de método, cuya herramienta de automatización es presentada en la sección 2.4. La sección 2.5 ahondará en ejemplos de aplicación práctica de ADM, ofreciendo una serie de lecciones aprendidas en estas experiencias. El capítulo finaliza con una

recapitulación de las principales conclusiones de este trabajo en la sección 2.6 y con la lista de referencias citadas en el texto en la sección 2.7.

2.2 Características de Ariadne Development Method (ADM)

ADM [14, 15] es un método de ingeniería para sistemas hipermedia y web que plantea un modelo de proceso iterativo y centrado en el usuario [30] con el que se pueden modelar las características del sistema de una forma completa, integradora e independiente de plataforma. Por un lado, el modelo de proceso es **iterativo y centrado en el usuario** con objeto de mejorar la *usabilidad* de las aplicaciones resultantes. Por otro, se establecen un conjunto de fases en las que se deben generar una serie de productos mediante los cuales se recogen todas las características del sistema, ya sean de navegación, estructurales, de presentación, de interacción, de acceso o de funcionamiento, como se discutirá en el apartado 2.2.2, de manera que se puede llevar a cabo un **modelado completo**. Además, dicho modelado es **integrador** en tanto en cuanto las diferentes interrelaciones existentes entre los productos generados son contempladas en forma de reglas de validación, que se encargan de verificar la completitud y la consistencia del diseño realizado. Finalmente, el modelado realizado es **independiente de plataforma** puesto que los diseños llevados a cabo con ADM pueden trasladarse a distintos entornos de operación. De hecho, ADM no es un método específico para web o para una herramienta de autor concreta, sino que persigue recoger las necesidades genéricas del proceso de desarrollo de los sistemas hipermedia, considerados como un superconjunto de los sitios web. Por ello, ofrece un conjunto de productos en los que cualquier referencia a la plataforma de especificación es obviada, puesto que ésta sólo debe influir al generar el prototipo o sistema. La inclusión de restricciones de implementación durante el diseño aumenta la provisionalidad de un método, puesto que ésta es una tecnología en constante y vertiginosa evolución y cualquier dependencia de tecnologías concretas hace que el método se quede obsoleto en poco tiempo. Con objeto de conseguir esta independencia de plataforma ADM hace uso de entidades de diseño que se fundamentan en un marco compuesto de dos niveles de abstracción:

- Las entidades de bajo nivel, o primitivas del diseño, son los componentes de cualquier sistema hipermedia, tales como nodos, contenidos, enlaces, anclas, relaciones estructurales, etc. Para definirlas se recurre a un modelo de referencia para hipermedia denominado Labyrinth [16, 18] que indica qué se entiende por cada uno de los elementos, qué estructura tienen, cómo se relacionan y cómo se pueden utilizar. Este modelo es independiente tanto del dominio de la aplicación como de la plataforma de implementación u operación, de forma que sus componentes pueden trasladarse a diversos entornos físicos.
- Las entidades de alto nivel son abstracciones realizadas sobre las primitivas del diseño para modelar las características del sistema con una notación sencilla que pueda ser discutida por los miembros de un equipo de desarrollo multidisciplinar y se corresponden con los productos de ADM [17]. Cada uno de esos productos aborda una o más perspectivas de diseño desde un determinado nivel de abstracción. Por ejemplo, el Diagrama Estructural de ADM es una representación

visual de la estructura lógica de la aplicación en el que participan primitivas del diseño como el nodo o las relaciones estructurales de agregación y generalización. Puesto que en estos productos no se incluyen restricciones de la plataforma de implementación, pueden materializarse en distintos entornos de operación.

Así pues, este método asume el modelo de referencia de hipermedia Labyrinth que determina los componentes básicos que pueden aparecer en un sistema hipermedia cualquiera. Sobre la base de este modelo de referencia se realizó un proceso de abstracción que dio lugar a una serie de entidades de diseño que son los productos de ADM. A continuación se estableció un proceso sistemático para generar dichos productos dando lugar al método en sí. Esta sección se inicia con una breve descripción de ese modelo de referencia, Labyrinth, en la sección 2.2.1. A continuación, se presentan algunas características relevantes del método en la sección 2.2.2 para, finalmente, describir el modelo de proceso adoptado en la sección 2.2.3.

2.2.1 El modelo de referencia de ADM: Labyrinth

Labyrinth [16, 18] es un modelo de referencia para hipermedia que define los componentes básicos de cualquier sistema hipermedia, sus relaciones estructurales y semánticas así como su comportamiento y funcionamiento. Según este modelo un sistema hipermedia o web tiene que ser un **hiperdocumento**, es decir, un sistema totalmente conexo, en el que hay un nodo central o eje desde el que existe un camino a todos los nodos del sistema desde los que, a su vez, también existe un camino que llega al nodo central [11]. La composición del hiperdocumento en Labyrinth está reflejada en la figura 2.1, donde se puede observar que éste se descompone en:

- Un Hiperdocumento Básico, equiparable con el sistema que se diseña para todos los usuarios, que accederán al mismo de acuerdo con las reglas que se establezcan, es decir, este hiperdocumento no sólo incluye los elementos de acceso público sino todos los componentes, tanto estáticos como dinámicos, del sistema.
- Un conjunto de Hiperdocumentos Personalizados, cada uno de los cuales pertenece a un usuario o a un grupo (representado en la figura con el concepto abstracto de sujeto) y en el que se guardan componentes que su propietario quiere mantener en un ámbito privado.

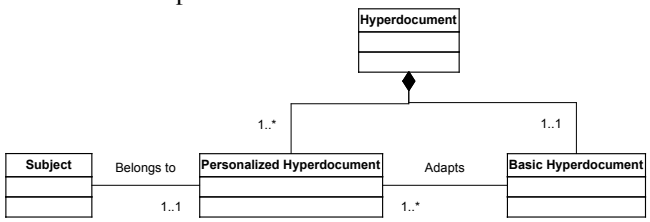


Fig. 2.1. Concepto de hiperdocumento en Labyrinth

Los usuarios pueden adaptar el contenido y la presentación de los nodos, así como la estructura del sistema, ya sea modificando sus elementos (enlaces, anclas, atributos o eventos) o bien creando otros nuevos. La visión personalizada del hiperdocumento constituye una forma de introducir información privada y ocultar elementos de contenido. La posibilidad de ajustar el hiperdocumento a las necesidades y

preferencias del usuario reduce considerablemente el riesgo de desorientación, haciendo la estructura hipermedia más manejable y familiar [16]. La personalización puede realizarse por parte de usuario individual o de grupo, dependiendo del alcance que se desee conceder a los cambios introducidos. La vista que el usuario tiene del hiperdocumento se construye superponiendo sus personalizaciones a las de su grupo, y éstas a la información del Hiperdocumento Básico. Así pues, los Hiperdocumentos Personalizados almacenan componentes procedentes del Hiperdocumento Básico que han sido adaptados por sus propietarios o bien nuevos elementos creados en ese espacio privado de trabajo.

Los componentes de un Hiperdocumento Básico, que se encuentran recogidos en la tabla 2.1, son siete: **usuarios**, **nodos**, **contenidos**, **anclas**, **enlaces**, **atributos** y **eventos**. Sobre estos elementos se establecen una serie de relaciones estructurales y semánticas (véase la tabla 2.2) que permiten ubicar contenidos en los nodos de manera que ambos componentes estén conceptual y físicamente separados (**Ubicación**), dotar de semántica y comportamiento e interactividad a los componentes del diseño (**Asignación de atributos** y **Asignación de eventos**, respectivamente) y establecer las reglas que rigen una acceso seguro y adecuado a la aplicación (**Asignación de reglas de acceso**). Labyrinth incluye un conjunto de operaciones que pueden realizarse sobre cada uno de sus componentes y relaciones, operaciones que a su vez pueden servir para programar los eventos, de manera que todo componente y relación es susceptible de definirse de forma procedimental [18].

Tabla 2.1. Elementos del Hiperdocumento Básico de Labyrinth

Elemento	Descripción
Usuario	Usuario individual o grupo. Los usuarios pueden crear grupos y definir vistas privadas para ellos o para su grupo.
Contenido	Elemento de información o de interacción de un determinado tipo (v.g., fragmento de texto, imagen, vídeo, animación, botón, campo de texto, etc.) que se caracteriza por contar con un espacio de representación formado por una serie de ejes de coordenadas (v.g. caracteres, palabras y frases para un texto). Los contenidos pueden ser simples o compuestos, en cuyo caso responden a una relación de agregación o de generalización. Además, los contenidos tienen una categoría de acceso que determina el tipo de operaciones que pueden realizarse sobre ellos.
Nodo	Contenedor abstracto de información que es una unidad conceptual. Los nodos tienen una dimensión espacial y otra temporal, en las que se pueden ubicar contenidos (con valores absolutos o relativos a través de relaciones espaciales y temporales). Los nodos pueden ser simples o compuestos, en cuyo caso responden a una relación de agregación o de generalización. Además, tienen una categoría de acceso que determina el tipo de operaciones que pueden realizarse sobre ellos.
Ancla	Referencia a una parte de un nodo, de un contenido o de un contenido contextual (contenido presentado en el contexto de un nodo específico) utilizada, fundamentalmente, para establecer enlaces. Las anclas también pueden enclavarse en todo el nodo o todo el contenido.
Enlace	Conexión entre dos conjuntos de anclas, el origen y el destino, que representa los caminos de navegación del sistema hipermedia así como las relaciones estructurales o de composición.

Atributo	Propiedad que puede asociarse a un usuario, nodo, contenido o enlace para incrementar su semántica.
Evento	Conjunto de acciones ejecutadas cuando se verifica una condición. El evento se asocia a nodos, enlaces o contenidos, de tal forma que se evalúa la condición cuando dicho elemento está activo. Es la base para la inclusión de especificaciones procedimentales (objetos virtuales creados en tiempo de operación) y el modelado de comportamientos interactivos.

Tabla 2.2. Relaciones entre los elementos del Hiperdocumento Básico de Labyrinth

Relación	Descripción
Ubicación	Permite colocar un contenido en algún lugar y/o momento dentro del espacio de representación de un nodo. Para ello pueden emplearse posiciones absolutas o relativas. En este último caso se puede hacer uso de una serie de relaciones espacio-temporales entre contenidos.
Asignación de atributos	Se emplea para incrementar la semántica de nodos, usuarios, enlaces y contenidos añadiéndoles meta-datos.
Asignación de eventos	Asocia eventos a nodos, contenidos y enlaces. La separación de los eventos con respecto a los elementos en que se ejecutan permite reutilizar el mismo evento en distintos contextos.
Asignación de reglas de acceso	Permite especificar las reglas de acceso al hiperdocumento siguiendo el modelo de seguridad presentado en [4]

2.2.2 Perspectivas de diseño

ADM ofrece artefactos de especificación para modelar seis perspectivas de diseño: la estructura, la presentación, la navegación, el comportamiento, el funcionamiento y el acceso. Cada una de estas perspectivas se aborda desde distintos niveles de abstracción pero, al mismo tiempo y merced a una serie de interrelaciones existentes entre los productos de ADM, de manera integrada. Para ello harán uso siempre de las mismas abstracciones que, además, pertenecen al dominio de la hipermedia, enfocándolas desde diferentes perspectivas. De esta forma, los desarrolladores pueden emplear el método para modelar de forma progresiva e integrada:

- las capacidades de navegación del sistema, utilizando productos como el Diagrama de Navegación o las Especificaciones de las Estructuras de Acceso;
- su estructura lógica, utilizando entre otros productos como el Diagrama Estructural o el Diagrama Interno del Contenido;
- su comportamiento, utilizando entre otros productos como el Catálogo de Eventos o las Especificaciones Detalladas de Funciones;
- las funciones que ofrece, utilizando entre otros productos como las Especificaciones Funcionales o las Especificaciones Detalladas de Funciones;
- las reglas de acceso, utilizando entre otros productos como las Reglas de Autorización o la Tabla de Acceso, y
- las características de la interfaz, utilizando entre otros productos como los Diagramas Internos o las Características de Presentación.

A modo de ejemplo ilustrador se comentará cómo especificar las capacidades de navegación, que incluyen tanto los enlaces como todas aquellas herramientas de

ayuda que se consideren oportunas. Esta especificación está representada en la figura 2.2, en la que se incluyen los distintos productos generados durante las dos etapas de diseño asumiendo un enfoque de diseño descendente o ascendente. En el primer caso se muestra mediante las líneas continuas y en el segundo mediante las discontinuas. A continuación se describe el enfoque descendente.

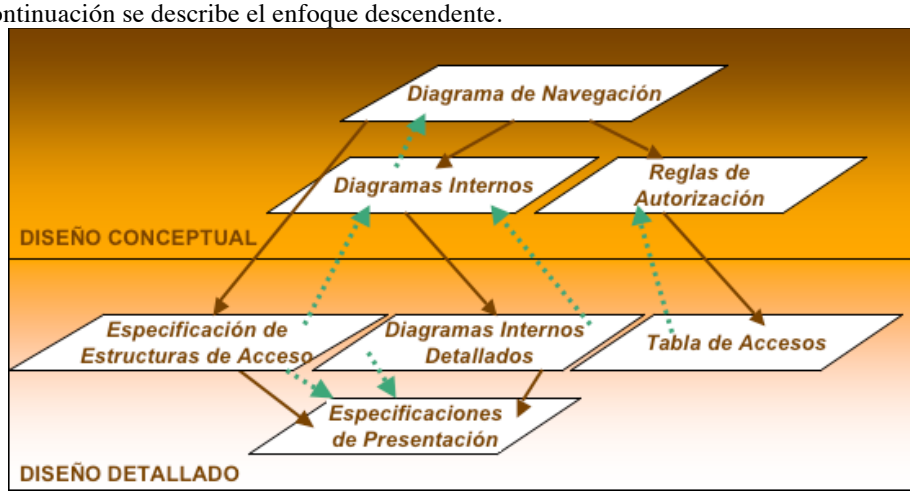


Fig. 2.2: Especificación progresiva e integrada de las capacidades de navegación mediante productos de ADM

La definición de los caminos y herramientas de la manera más abstracta se inicia durante el Diseño Conceptual al generar el Diagrama de Navegación, en el que se identifican enlaces-tipo entre nodos-tipo y herramientas de navegación tipo. Por ejemplo, en un sitio de comercio electrónico un enlace-tipo "Descripción Detallada" conectaría la breve reseña de un producto que aparecería en el catálogo con otra más exhaustiva. Una herramienta de navegación abstracta sería el propio catálogo que, de momento, sólo aparece como tal sin indicar qué tipo de herramienta se implementará (un índice estático o dinámico, personalizado o no, un mapa, etc.).

El siguiente nivel de concreción se alcanza en la misma fase del desarrollo al generar los Diagramas Internos de Nodos y de Contenidos, en los que se crea una plantilla de los nodos-tipo y de las herramientas de navegación-tipo. Estas plantillas son una representación de la interfaz de los mismos, junto con su semántica (es decir, los atributos que tienen asociados) y su comportamiento (es decir, los eventos que están ligados a ellos). Siguiendo con el caso del sitio web de comercio electrónico, en el Diagrama Interno correspondiente al nodo-tipo "Producto" se incluirá un contenido-tipo o "caja" que actúe como origen del enlace "Descripción detallada". Si, por ejemplo, se deseara que dicha descripción se adaptara a ciertas características del usuario como por ejemplo la edad, haciendo uso de la técnica denominada presentación adaptativa [10], de tal forma que a cada usuario se le presente el texto más adecuado, se asociaría un evento al enlace al que se añadiría una descripción abstracta de la regla de personalización. Asimismo, el Diagrama Interno del "Catálogo" recogería un esbozo de la forma que tendrá este nodo y si es dinámico o personalizado tendría asociado el correspondiente evento. Las descripciones de los eventos, a este nivel de abstracción, se pueden hacer de manera informal.

Al mismo nivel de abstracción se encuentran las Reglas de Autorización que contienen las reglas de acceso que determinan qué puede hacer cada usuario y qué hiperdocumento hay que generar para él. Estas reglas se pueden emplear para establecer qué enlaces y herramientas de navegación pueden ser vistos por cada tipo de usuario o rol, de tal forma que cuando un usuario no puede activar un enlace, éste no se le presenta como tal. Así por ejemplo, podría existir un enlace "Productos favoritos" dentro del nodo "Catálogo" sólo visible para usuarios registrados, lo cual supondría la inclusión de una regla de acceso del tipo *<inclusión de favoritos, usuario registrado>*, de manera que otro tipo de usuario no pueda utilizar esta opción.

Un peldaño más abajo en el nivel de abstracción lleva al Diseño Detallado, en el que se identifican instancias concretas y se producen especificaciones más completas. En el ejemplo utilizado, el "Catálogo" sería completamente descrito de manera que las descripciones informales se sustituyen por un lenguaje formal o por código. Si, por ejemplo, el "Catálogo" es dinámico se incluiría la regla o consulta que lo calcula y si, además, es personalizado, en dicha regla se concretaría cómo adaptar los contenidos de acuerdo con el modelo de representación del conocimiento que se haya adoptado. En los Diagramas Internos Detallados, los contenidos-tipo de la fase anterior se sustituyen por contenidos concretos (ya sean contenidos embebidos o referencias a contenidos). Así, la "caja" correspondiente al origen del enlace "Productos favoritos" del nodo "Catálogo" se sustituiría, por ejemplo, por un icono específico. Además, las reglas de acceso genéricas se particularizan a nodos, contenidos y roles concretos a través de la Tabla de Acceso. En este caso, la regla *<inclusión de favoritos, usuario registrado>*, haría que ser generara una prohibición de que cualquier usuario que no esté registrado pueda ver el contenido-tipo ("Productos favoritos"), que es el origen del enlace que permite utilizar esa opción.

Finalmente, se indican las Características de Presentación de cada nodo y contenido concreto. Por ejemplo, se podría establecer el tamaño del botón "Descripción Detallada" que podría depender de la plataforma de uso o si se presenta sólo el icono, el icono y la leyenda asociada o sólo ésta última dependiendo de algún parámetro.

2.2.3 Modelo de proceso

El proceso de desarrollo propuesto en ADM (véase la figura 2.3) se caracteriza por ser sistemático, en la medida en que está perfectamente definido a través de una serie de fases, cada una de las cuales se descompone en un conjunto de actividades que tienen como objetivo la creación de uno o más productos.

Fig. 2.3.: El proceso de desarrollo en ADM. Las cajas son las fases del método y las flechas representan interrelaciones entre las mismas.

En su estado actual ADM no cubre las etapas de estudio de factibilidad ni de análisis, presuponiéndose en todo momento que se parte de una especificación de requisitos. El proceso aquí propuesto consta de tres fases: Diseño Conceptual, Diseño Detallado y Evaluación, cada una de las cuales se descompone en una serie de actividades que dan lugar a unos productos (véase la tabla 2.3). Hay tres aspectos de este proceso de desarrollo propuesto por el método que conviene resaltar.

En primer lugar, cada una de las fases de ADM encara el desarrollo desde un nivel de abstracción distinto a la vez que complementario. Mientras el Diseño Conceptual tiene como objetivo describir las características estáticas y dinámicas del sistema desde una perspectiva muy abstracta, en términos de lo que suelen denominarse entidades-tipo [28], el Diseño Detallado da lugar a una especificación muy concreta casi trasladable automáticamente a unidades de implementación y, finalmente, la Evaluación hace posible verificar la validez y utilidad de las soluciones propuestas, ya sea en forma de productos de diseño o de prototipos, involucrando al usuario o destinatario en este proceso. Por ejemplo, durante el diseño de un sitio web de comercio electrónico en la fase de "Definición de la estructura lógica" del Diseño Conceptual (véase la tabla 2.3) se indica la forma en que se organiza la información a través del Diagrama Estructural. En este diagrama hay una serie de nodos abstractos o nodos-tipo, como podrían ser el "Catálogo", el "Producto" o el "Carrito de la compra". Todas estas entidades se dice que son abstractas, o entidades-tipo, puesto que cada una de ellas representa a varias instancias concretas del sistema final. Dichas instancias, o nodos concretos, se especifican durante el Diseño Detallado en el que se crearán tantas réplicas del nodo-tipo "Producto" como sea necesario, con la ventaja de que todos ellos tendrán unas características comunes especificadas en el nodo-tipo. Además, las tres fases son independientes, puesto que cada una de ellas puede llevarse a cabo sin necesidad de haber pasado por las restantes. Sin embargo, todas ellas están interrelacionadas, pues en conjunto ofrecen una visión global del sistema. Dichas relaciones vienen representadas por las flechas de la figura 2.3. Como se puede observar, entre el Diseño Conceptual y el Detallado esa relación está mediada por una serie de reglas de validación y verificación orientadas a garantizar la completitud, la integridad y la consistencia del diseño y de cada uno de sus productos. Dichas reglas se definen tanto a nivel de producto específico (intra-reglas) como entre distintos productos de la misma o de distinta fase (inter-reglas).

En segundo lugar, ADM no establece una secuencia entre las fases, sino un proceso flexible que será llevado a cabo como el equipo de desarrollo decida. Esto permite que se pueda realizar un diseño descendente que comience con el Diseño Conceptual, proceda a continuación con el Diseño Detallado y finalice con la Evaluación. Asimismo, si se requiere un proceso de *prototipado* rápido [Preece *et al*, 2002], como suele ocurrir en la mayoría de los desarrollos de sitios web, se puede adoptar un enfoque ascendente que se inicie creando un prototipo (actividad ligada a la fase de Evaluación) para después pasar al Diseño Detallado o al Conceptual cuando sea posible destinar recursos a dichas fases. También se puede aplicar un enfoque mixto en el que haya distintos ciclos, unos ascendentes y otros descendentes.

Finalmente, se plantea un proceso iterativo y centrado en el usuario que, a través de la fase de Evaluación, toma parte activa en el desarrollo. En la evaluación puede analizarse un prototipo, un sistema o cualquier producto generado durante el Diseño Conceptual o Detallado, con objeto de contrastar la utilidad de las soluciones propuestas y detectar posibles deficiencias así como mejoras. Los resultados de la Evaluación realimentan el resto de las fases hasta conseguir el producto adecuado.

Tabla 2.3. Fases, actividades y productos de ADM. El orden empleado no es la secuencia de realización.

Fase	Actividad	Productos
Diseño Conceptual	Definición de la estructura lógica	• Diagrama Estructural
	Estudio del funcionamiento del sistema	• Diagrama de Navegación • Especificaciones Funcionales
	Especificación de entidades	• Diagramas Internos de Nodos • Diagramas Internos de Contenidos • Catálogo de Atributos • Catálogo de Eventos
	Modelado de Usuarios	• Diagrama de Usuarios
	Definición de la política de acceso	• Catálogo de Categorizaciones • Reglas de Autorización
Diseño Detallado	Identificación de instancias	• Instancias de Nodos • Diagrama de Usuarios Instanciado
	Especificación de funciones	• Especificaciones de Estructuras de Acceso • Especificaciones Detalladas de Funciones
	Especificación de instancias	• Diagramas Internos Detallados de Nodos • Diagramas Internos Detallados de Contenidos • Reglas de autorización • Asignación de usuarios
	Diseño de la presentación	• Especificaciones de Presentación

Evaluación	Desarrollo del prototipo	• Prototipo
	Realización de la evaluación	• Documento de la Evaluación • Informe de Conclusiones

2.3 Fases y productos de ADM

En esta sección se describen las tres fases de ADM utilizando como ejemplo la aplicación de este método al diseño de Now-Graduado [2], un CD-ROM educativo implementado con MacroMedia Director dentro del marco del proyecto europeo NOW-META. Ésta fue una de las primeras experiencias de desarrollo sistemático de sistema hipermedia abordada por los autores de este capítulo y permitió, junto con otras muchas (véase la sección 2.5), detectar necesidades de modelado específicas que no podían ser cubiertas con otros métodos.

2.3.1 Diseño Conceptual

La fase de Diseño Conceptual de ADM tiene como objetivo desarrollar una especificación de la estructura y del funcionamiento del sistema desde un punto de vista abstracto, es decir, identificado entidades-tipo. No se busca pues definir los contenidos o enlaces concretos, sino que se trata de iniciar el diseño desde un nivel de abstracción más alto. Por ejemplo, en Now-Graduado nodos-tipo pueden ser "Ejercicio" o "Curso", contenidos-tipo "Definición" o "Pregunta" y usuarios-tipo "estudiante".

Las actividades realizadas en esta fase, recogidas en la tabla 2.3, se comentan a continuación. Al igual que en el caso de las fases del método, no existe un orden predefinido, si bien en un enfoque descendente podría aplicarse tal y como se describe aquí. Esta fase está soportada por una herramienta de automatización denominada AriadneTool, descrita en la sección 2.4.

Definición de la estructura lógica. Esta actividad tiene como objetivo representar de forma gráfica la estructura del sistema a través del *Diagrama Estructural* que debe generarse en todo desarrollo para plasmar la organización de la información. Como ejemplo, la Figura 2.4 muestra el *Diagrama Estructural* de Now-Graduado, en el que se indica que el sistema está formado por cinco partes: la presentación del curso; la ayuda, estructurada en un conjunto de temas; el glosario de términos; el curso en sí y un sistema externo que almacena y procesa la información relacionada con las sesiones de estudio del alumno. Un curso se divide en módulos, que a su vez comprenden lecciones consistentes en varias páginas de temas teóricos, ejercicios y actividades para saber más sobre el aspecto tratado. Para los ejercicios se definieron varias estrategias de resolución, entre las que se encuentran las recogidas en la figura.

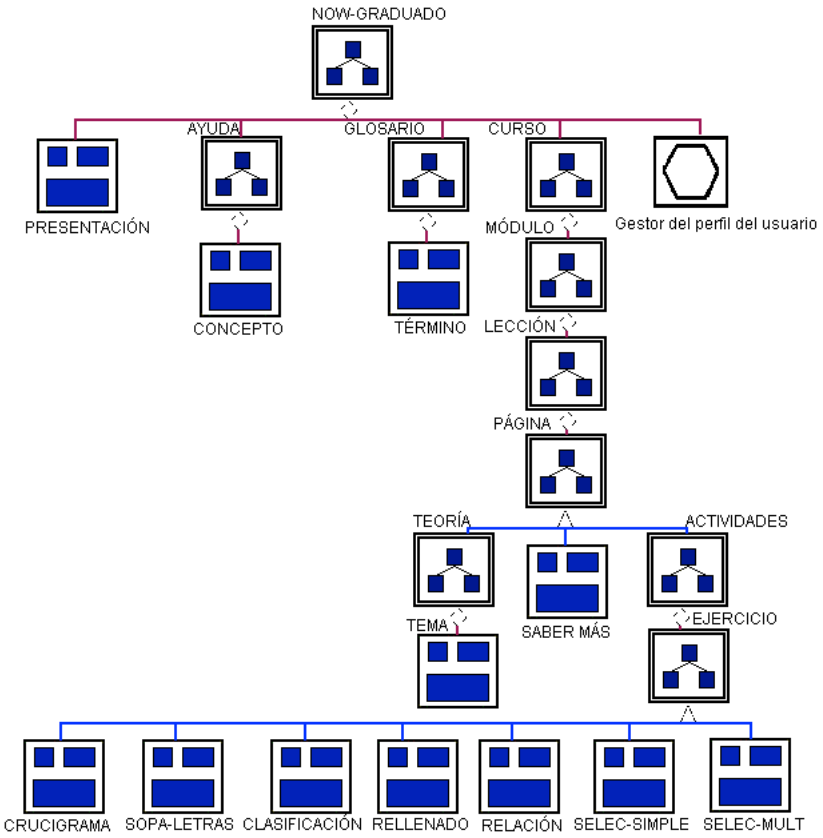


Fig. 2.4.: Diagrama Estructural de Now-Graduado. El método hace uso de la notación de UML [9] para representar las agregaciones y las generalizaciones.

La estructura se especifica por medio de nodos-tipo, que pueden ser simples o compuestos, y de dos mecanismos de abstracción: la generalización y la agregación. Además se pueden incluir referencias a sistemas externos que no son objeto del desarrollo a través de lo que se denominan Elementos Externos (por ejemplo, el "Gestor del Perfil del Usuario" en la figura 2.4).

Un nodo simple es un contenedor de información que se trata como una unidad de recuperación que siempre tiene existencia física y que puede instanciarse durante el Diseño Detallado para producir los nodos finales. Por ejemplo los nodos "Concepto" o "Ejercicio" de la figura 2.4 darán lugar a diversas instancias que representan todos los temas tratados y las actividades propuestas a los alumnos, respectivamente.

El nodo compuesto es un mecanismo de abstracción con el que representar la estructura del hiperdocumento y, como tal, no siempre tiene una existencia física. Todo nodo compuesto debe ser origen de al menos una relación de generalización o de agregación cuyo destino serán uno o más nodos simples o compuestos. Una agregación hace posible hacer referencia a un conjunto de nodos por medio de un único elemento, el objeto compuesto. Se trata de una relación de composición débil, en tanto en cuanto los elementos permanecen independientes a todos los efectos [21].

Así por ejemplo, en la figura 2.4 podemos ver que una "Lección" está formada por un conjunto de "Páginas". La relación de generalización es una composición con una semántica más fuerte, puesto que los componentes pertenecen a una misma categoría y como tal comparten una serie de propiedades y comportamientos. Al definir una relación de este tipo, se activan los mecanismos de herencia, de tal forma que todo aquello que se asocie al elemento generalizador en el Diagrama Interno, en cualquiera de los Catálogos o en el Diagrama de Navegación es heredado inmediatamente por los objetos generalizados. Por ejemplo, en la figura 2.4 el nodo "Página" se especializa en tres tipos de nodos: "Teoría", "Ejercicios" y "Saber Más".

Estudio del funcionamiento del sistema. Durante esta actividad se analizan los servicios que se van a proporcionar al usuario, distinguiéndose entre dos tipos de funciones: facilidades de navegación, representadas en el *Diagrama de Navegación* y otras funciones no relacionadas con la navegación, recogidas por medio de las *Especificaciones Funcionales*.

El *Diagrama de Navegación* incluye enlaces entre nodos o elementos externos así como herramientas de navegación. La figura 2.5 muestra las posibilidades de navegación de Now-Graduado que desde el primer nodo, "Presentación", permite acceder a la ayuda (si es la primera sesión del alumno) o al "Curso" (en caso contrario). Una vez en éste, se puede estudiar, en cuyo caso se presenta el tema en el que se quedó el alumno, o repasar, en cuyo caso se accede al índice de contenidos en el que se marcan los temas ya estudiados. Desde cualquier página se puede acceder al Glosario de dos formas: si se selecciona un término resaltado en la página se abre una ventana emergente con su definición y si se selecciona el icono "Glosario" se accede al índice alfabético del mismo. Existe una secuencia entre las páginas, materializada por el enlace "Anterior/Siguiente", aunque también se puede navegar a páginas relacionadas a través de las correspondientes asociaciones o enlaces hipertextuales. Desde cada página se puede ir también al nodo "Saber más", que incluirá actividades e información adicional para ese tema, así como a los ejercicios. Finalmente, existe un enlace "Inicio" desde el que se puede acceder al comienzo de la aplicación desde cualquier otro del sistema.

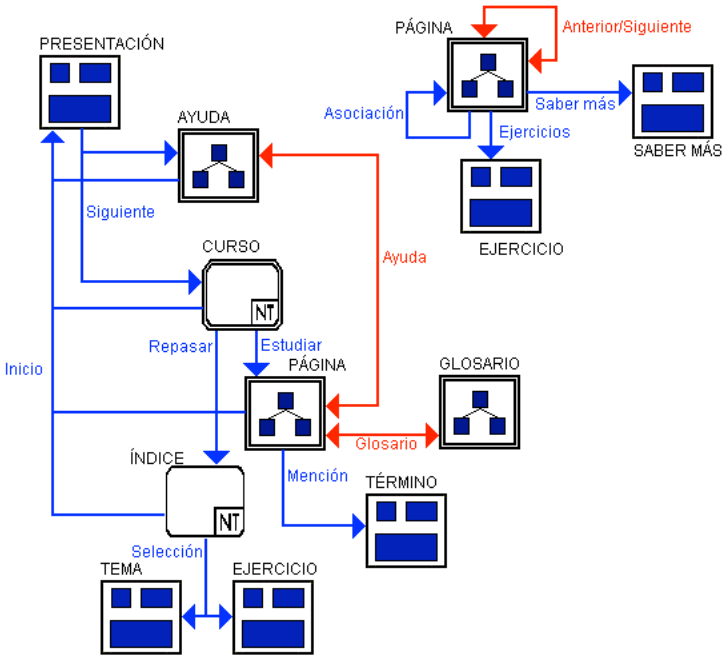


Fig. 2.5.: Diagrama de Navegación de Now-Graduado.

Los enlaces están etiquetados, pueden tener asociados atributos que incrementen su semántica [32] y pueden ser unidireccionales (véanse los enlaces "Ejercicios", "Asociación" o "Siguiente" en la figura 2.5) o bidireccionales (véanse los enlaces "Anterior/Siguiente", "Ayuda" o "Glosario" en la figura 2.5). Además, pueden incluirse enlaces n-arios con varios orígenes o destinos, como por ejemplo los enlaces: "Siguiente", que se utiliza para determinar qué nodo debe activarse al terminar la presentación del CD, el "Inicio" que permite definir un único enlace multi-origen o "Selección" que permite decidir entre dos posibles destinos. Como se puede ver en la figura, el enlace "Inicio" parte del nodo general "Página" y, en consecuencia, parte de cada una de sus especializaciones (los nodos "Teoría", "Saber Más" y "Actividades").

Además, pueden incluirse herramientas de navegación en forma de nodos simples o compuestos. Estas herramientas pueden corresponderse con nodos que aparecían ya en el Diagrama Estructural y de los que ahora se indican sus características de navegación. Así por ejemplo, muchas agregaciones suelen convertirse en una herramienta de navegación que permite acceder a los elementos que componen el nodo compuesto, como por ejemplo el "Curso" de la figura 2.5. También pueden ser nuevos nodos que haya que incluir porque sólo tienen como objeto facilitar la navegación, como por ejemplo el nodo "Índice".

Las *Especificaciones Funcionales* tienen como objetivo modelar las funciones que se van a ofrecer a los usuarios y que no están relacionadas con la navegación por el hiperdocumento (v.g., impresión, edición de contenidos, cálculos, creación dinámica de objetos...). Dentro de estos servicios se pueden considerar los siguientes grupos:

- Funciones de personalización del hiperdocumento: permiten que los contenidos y la estructura se adapten a las necesidades de cada usuario. En su versión más simple, puede consistir en la creación de notas personales, y en la más compleja, la posibilidad de definir sistemas adaptativos.
- Funciones de mantenimiento del hiperdocumento: hacen posible la modificación de los contenidos y/o la estructura del documento.
- Funciones de comunicación entre los usuarios: permiten que los usuarios se comuniquen, de forma sincrónica o a sincrónica, por medio de mensajes multimedia.
- Funciones de interfaz de usuario: permiten incluir funciones que mejoren la usabilidad del sistema.
- Otros servicios: por ejemplo, impresión de páginas, exportar nodos y sus contenidos en algún formato, etc.

Esta especificación funcional se realiza utilizando una técnica de diseño descendente, en la que cada función puede descomponerse en otras más sencillas. Además, una función puede ser iniciada por un evento (véase la definición en la tabla 2.1) y el mismo evento puede dar lugar a la activación de varias funciones. Ejemplos de funciones en el caso de aplicación de Now-Graduado son "Iniciar sesión de estudio", que decide qué hay que hacer cuando un usuario entra (es decir, calcula el destino del enlace multi-destino "Siguiente") o "Corregir ejercicio", que determina si la actividad se ha resuelto correctamente y envía los datos del alumno al "Gestor del Perfil del Alumno".

Especificación de entidades. En esta fase se detallan los nodos identificados hasta el momento y los contenidos que se van a incluir en los mismos. Los resultados que se producen en esta fase son los *Diagramas Internos* de cada nodo y contenido y los *Catálogos de Atributos* y de *Eventos*. Todos los productos de esta fase son obligatorios, pues un sistema hipermedia sin atributos o eventos sería un pobre ejemplo de sistema, tan sencillo que tal vez el uso de un método no estaría justificado.

El *Diagrama Interno de los Nodos* se divide en dos partes: el Diagrama Espacial en el que se describe al nodo en el espacio bidimensional de la pantalla y la Línea de Tiempo, en la que se define la evolución del nodo a lo largo del tiempo. Los contenidos pueden ubicarse en el nodo, tanto en su Diagrama Espacial como en la Línea de Tiempo, dependiendo de su naturaleza. Por ejemplo, se podría establecer que un texto aparezca en un determinado sitio en el momento t y desaparezca x segundos después. Además, las ubicaciones pueden hacerse utilizando un valor concreto (como en el ejemplo anterior) o haciendo depender éstas de una relación espacial o temporal con otro contenido. En este caso se hará uso relaciones denominadas alineaciones y sincronizaciones propias de Labyrinth [18].

Las alineaciones son arcos multi-destino en los que los destinos se ubican en una posición relativa con respecto a la posición de la del origen. La relación espacial se expresa utilizando tres tipos de parámetros: la topología, la dirección y la distancia, pudiéndose dar valor a uno de ellos o a varios.

- La topología está basada en el modelo de las 4-intersecciones definido en [20] que emplea las relaciones entre los bordes y los interiores de figuras planas. Se ha hecho uso de aquellas relaciones topológicas que tienen sentido teniendo en cuenta que se considera que los contenidos son áreas rectangulares y que incluyen los operadores disjunto, junto, solapado, cubre, contiene e igual.

- La dirección indica en qué región debe estar el destino con respecto a donde se encuentra el origen: arriba, debajo, izquierda, derecha, arriba- izquierda, arriba-derecha, debajo - izquierda, debajo - derecha.
- La distancia es la distancia relativa entre las dos esquinas superiores del origen y del destino en el eje X e Y. Si se da valor a ambos parámetros, no tiene sentido especificar la topología o la dirección. Sin embargo, puede no darse valor a la distancia en ambos ejes y usar los otros dos parámetros para definir relaciones espaciales más complejas. Por ejemplo, una relación <*Contenido1, <junto, izquierda, (-,0)>, Contenido2*> significa que el *Contenido2*está alineado por el borde superior con el *Contenido1* y, además, está pegado a él.

Para simplificar la definición de alineaciones, ADM ofrece algunos valores predefinidos para las más frecuentes, como por ejemplo, alineación por el borde superior, por el inferior, por el derecho y por el izquierdo. En caso de necesitar relaciones más complejas el diseñador puede hacer uso de los tres parámetros previamente explicados. Una alineación se caracteriza porque cuando el origen de la misma se desplaza, los destinos del arco se moverían para mantener la restricción estipulada. A su vez, los destinos del arco no podrían moverse si con su desplazamiento violasen la restricción espacial expresada en la alineación.

Las sincronizaciones son también arcos multi-destino en los que los destinos aparecen o terminan en un momento que depende del inicio o del fin del contenido origen del arco. En este caso el arco se etiqueta con un operador temporal basado en los definidos en [6]. Así se puede establecer:

- que el origen aparezca *t* unidades de tiempo antes de que aparezcan los destinos (operador *antes(t)*);
- que los destinos se muestren a continuación del origen (operador *seguido()*);
- que el origen se muestre durante la presentación de los destinos (operador *durante(t)*, siendo *t* el tiempo que transcurre desde el final del contenido origen del arco hasta el inicio de los destinos del arco);
- que los destinos empiecen a mostrarse un tiempo *t* antes de que acabe el origen (operador solapa(t)), y, finalmente
- que todos empiecen o terminen a la vez (operadores *inicio_paralelo()* y *fin_paralelo()*, respectivamente).

El uso de operadores temporales y espaciales proporciona un poderoso mecanismo para especificar restricciones que pueden también ser empleado como condición de activación de los eventos. Esta aproximación es pues más rica que la aplicada en el modelo de referencia Amsterdam [22] puesto que los modelos espaciales y temporales asumidos tienen mayor semántica y riqueza expresiva [15].

En los Diagramas Internos de los Nodos también hay que especificar las anclas de los enlaces que parten o llegan a ese nodo. Para ello, si se trata de todo el nodo (especialmente habitual cuando se trata de marcar el destino de los enlaces) sólo se le asocia el enlace y si se trata de una zona específica o de un contenido concreto se marca éste con un patrón distinto para distinguirlo de aquellos que no actúan ni como origen ni como destino de un enlace. Además de contenidos, los nodos pueden tener asociados meta-datos o propiedades en forma de atributos así como eventos que describan su comportamiento. Ambos elementos se mantienen en dos catálogos, por lo que aquí tan sólo se establece una relación de asignación.

El *Diagrama Interno de Contenidos* es básicamente igual aunque también incluye un Diagrama Estructural para poder especificar contenidos compuestos. Por ejemplo, una barra de botones es una agregación de contenidos.

El *Catálogo de Atributos* es el repositorio de atributos en el que se almacenan todas las propiedades junto con su valor por defecto, valor que puede ser modificado al ser asignado a un elemento concreto

En el *Catálogo de Eventos* se almacenan los eventos que pueden ser reutilizados en distintos objetos. Un evento tiene un identificador que permite hacer referencia al mismo, un nombre representativo de su función, una condición de activación, unas operaciones que se ejecutan cuando dicha condición se activa y unas funciones relacionadas (véase arriba la descripción del producto *Especificaciones Funcionales*). La condición del evento se evalúa siempre que esté activo cualquiera de los elementos a los que se asocia, que pueden ser nodos, contenidos o enlaces.

Como ejemplo de esta actividad se describe, a continuación, cómo se modeló uno de los tipos de ejercicios interactivos de Now-Graduado: el crucigrama que se muestra en la figura 2.6. El ejercicio se compone de un enunciado, un alfabeto interactivo y la cuadrícula del crucigrama. El enunciado indica al estudiante qué palabras se están buscando y para escribirlas éste tendrá que hacer uso del alfabeto. Para escribir una letra deberá arrastrase con el ratón hasta una casilla, donde al soltarla permanecerá escrita. Si la letra no se suelta en una casilla, entonces se borra.

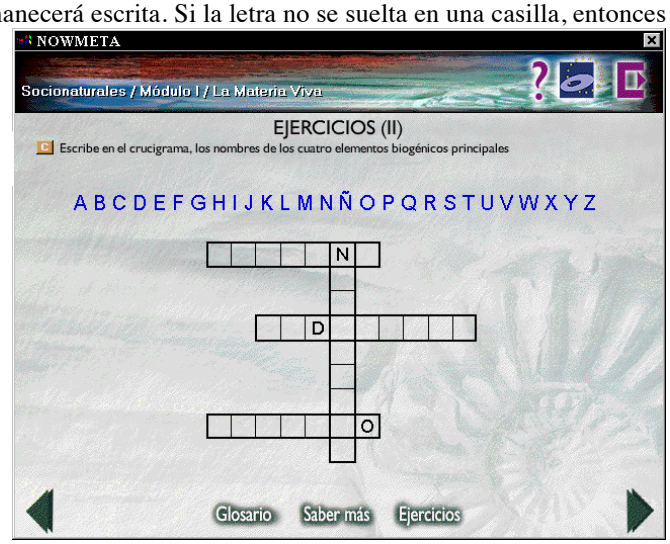


Fig. 2.6.: Ejemplo de ejercicio interactivo de Now-Graduado

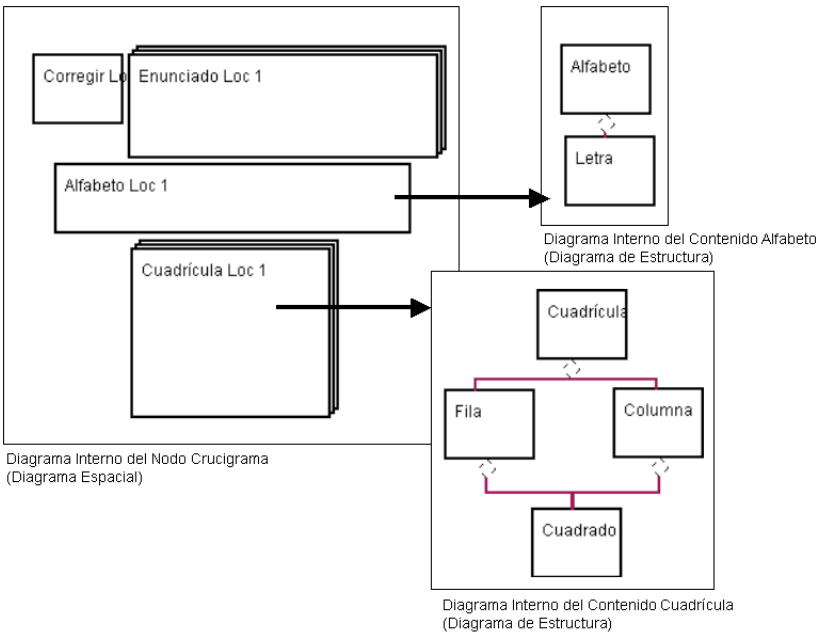


Fig. 2.7.: Ejemplo de Diagrama Interno de Nodo y de Diagramas Internos de Contenido. En la imagen sólo se aprecia la organización de los contenidos del nodo en el espacio bidimensional de la pantalla así como la estructura de los contenidos compuestos.

Durante el Diseño Conceptual se hizo una especificación genérica de este tipo de ejercicio que pudiera reutilizarse para generar en el Diseño Detallado tantos crucigramas concretos como hiciera falta con un mínimo coste. Para ello el nodo "Crucigrama" incluye cuatro contenidos: el botón de corrección, el enunciado, el alfabeto y la cuadrícula (véase la parte izquierda de la figura 2.7). En el Diagrama Interno del Nodo "Crucigrama" aparecen dos tipos de contenidos: los mono-instancia (caja única en la figura), que son aquellos que asumen el mismo valor en todas las instancias de ese nodo y los multi-instancia (cajas apiladas en la figura), que son los que pueden asumir distintos valores en distintas instancias del nodo. En la figura 2.7 sólo se muestra la parte del Diagrama Espacial del nodo "Crucigrama" que es interesante para el ejemplo, habiéndose obviado elementos no significativos como las barras de botones. Tanto el alfabeto como la cuadrícula son dos agregaciones de letras y cuadrados, respectivamente (véase la parte derecha de la figura 2.7). Cada letra tiene asociado un evento que permite gestionar su movimiento por la pantalla, evento que hace uso de la relación espacial <*contiene*, -, -> entre el cuadrado y la letra para decidir qué hacer cuando el usuario suelta una letra que estaba arrastrando. A su vez, cada cuadrado tiene asociado otro evento con el que se permite escribir la letra en él y dos atributos que representan el valor correcto que se debería escribir y el que se ha escrito, de manera que dichos atributos puedan ser empleados por el proceso de corrección que, a su vez, es iniciado cuando el contenido "Corrección" es seleccionado por el usuario. Este ejercicio tiene un elevado grado de interacción y supuso el uso de varios eventos, algunos de los cuáles se asignaban de forma

dinámica como puede verse en [18]. De esta forma, el funcionamiento del ejercicio queda especificado de manera que para definir las distintas instancias bastará con dar valor al enunciado y a la cuadrícula del crucigrama.

Modelado de usuarios. Esta tarea consiste en identificar los tipos de usuarios que tendrá el sistema. Como resultado se genera el **Diagrama de Usuarios** que es una pieza fundamental para el modelado de la política de acceso del sistema. Podría no ser necesario realizar este producto si el sistema a desarrollar no ofrece ni perfiles ni diversas posibilidades de acceso, como ocurría en el caso de Now-Graduado. En el capítulo 7 ("Modelado del acceso y de la seguridad") se profundiza en el uso de ADM para modelar el acceso, por lo que es allí donde se incluye un ejemplo detallado de este producto y de los que se generan en la actividad "Definición de la política de Acceso".

Puesto que se está en la fase de Diseño Conceptual el objetivo es identificar tipos de usuarios no usuarios concretos. Además, se asume un modelo de acceso basado en roles, o RBAC, que resulta más eficiente que uno basado en grupos [23]. Un rol es un puesto organizativo o responsabilidad que existe en el dominio de la aplicación mientras un equipo es sólo un grupo de usuarios. Los roles y los equipos soportan mecanismos de composición para poder tratar con estructuras complejas de usuarios. Así, es posible definir jerarquías de roles en forma de un Grafo Acíclico Dirigido (GAD), en el que los roles generales se especializan en otros más específicos utilizando una relación de generalización que implica herencia. A su vez, los equipos se definen como agregaciones de roles y equipos, es decir, mantienen una relación de composición que hace posible referirse de manera global a distintos sujetos manteniendo la independencia entre los componentes.

Definición de la política de acceso. El objetivo de esta tarea es definir las reglas que se van a aplicar al utilizar la aplicación en términos del tipo de actividades que se van a permitir a cada tipo de usuario. A tal efecto se emplea el modelo presentado en [Aedo *et al*, 2003], que se materializa a través de dos productos:

- El *Catálogo de Categorizaciones*, donde se asocia a cada objeto (nodo y contenido) una categoría de seguridad que determina la operación más permisiva que se le puede aplicar.
- Las *Reglas de Autorización*, en la que se especifica qué funciones, de aquellas incluidas en las Especificaciones Funcionales, pueden ser iniciadas por cada tipo de usuario.

Al igual que ocurría con el *Diagrama de Usuarios*, si no se van a ofrecer distintas vistas del sistema a distintos tipos de usuarios esta actividad no se llevaría a cabo.

En el *Catálogo de Categorizaciones* se asocia a todos los objetos, es decir, a los nodos y a los contenidos, una categoría que determina el tipo de operaciones que se van a poder aplicar sobre ellos. Para ello, se hace uso de tres valores sobre los que se define una relación de orden parcial ($\subseteq$), que verifica pues las relaciones reflexiva, antisimétrica y transitiva y según la cual cada categoría añade permisos a la anterior:

- Navegación ("Browsing"): categoría que sirve para recuperar información (nodos y contenidos), bien seleccionando enlaces o usando otros medios tales como índices, mapas o motores de búsqueda.
- Personalización ("Personalizing"): categoría que añade a la anterior la posibilidad de incluir elementos personales (como por ejemplo, contenidos privados, nodos o

enlaces), esto es, la posibilidad de crear y actualizar espacios privados o personalizados, ya sea a nivel individual o de grupo.

- Edición ("Editing"): categoría que añade a la anterior la posibilidad de modificar elementos, haciendo que dichos cambios sean visibles para todos los usuarios.

Así, si se decide que un objeto es del tipo "Browsing" nadie, independientemente de los permisos que tenga asignados podrá modificarlo.

Además, en las *Reglas de Autorización* a cada sujeto, es decir, a cada rol y equipo, se le puede habilitar para ejecutar una serie de funciones.

Dado que tanto los sujetos como los objetos se definen mediante estructuras jerárquicas, existen unas reglas de propagación de autorizaciones [4] que facilitan al diseñador la especificación de las reglas de acceso:

- Herencia: todo elemento hereda de su padre los permisos de acceso.
- Sobreescritura de la herencia: el permiso heredado puede modificarse en el hijo.
- Herencia múltiple en paralelo: si un elemento tiene varios padres, se hereda la regla más permisiva.
- Herencia múltiple en secuencia: si un elemento tiene varios antecesores, se hereda del antecesor inmediato
- Asignación directa por agregación: si no hay otra opción, se asume la categoría de la agregación a la que pertenece el elemento.
- Múltiple asignación en paralelo: si no hay otra opción y un elemento pertenece a varias agregaciones, se asigna la regla más permisiva de los elementos que le agregan.

2.3.2 Diseño Detallado

Esta fase se centra en un nivel de abstracción más bajo que el Diseño Conceptual, pues aquí se utilizan entidades muy cercanas a las unidades de implementación. Se pasa de entidades-tipo a entidades concretas definidas de forma declarativa (por ejemplo, con una *uri*, *url* o referencia a un fichero) o procedimentalmente (por ejemplo, con un "*script* "o una consulta a una base de datos). Su gran dependencia de la plataforma técnica hace que no se pueda abordar con tanto detalle como la fase descrita en la sección anterior, pues se corre el riesgo de dejarse influenciar por el estado actual de una tecnología en constante evolución.

Durante el Diseño Detallado se producen instancias concretas de las entidades que conforman la aplicación, desde el punto de vista de su estructura lógica y de navegación, de la presentación, del funcionamiento, del comportamiento y del acceso. Si se aplica un enfoque descendente, en esta fase se definirán instancias de las entidades-tipo del Diseño Conceptual, mientras que desde un enfoque ascendente se abstraerían propiedades y comportamientos del prototipo generado durante la Evaluación.

Las actividades a llevar a cabo se resumen en la tabla 2.3. En la descripción que sigue se adopta un enfoque descendente.

Identificación de las instancias. En esta actividad se generan instancias concretas de las entidades del sistema que se identificaron en el Diseño Conceptual, de tal manera que se crean réplicas de estructuras o elementos que pueden reutilizarse en diferentes contextos. De hecho se van a producir instancias de dos tipos:

· Instancias de elementos compuestos: lo que supone generar réplicas de toda la estructura que se inicia en ese elemento.
· Instancias de elementos simples: lo que implica producir copias de ese elemento.

En esta actividad se generan dos productos: el *Diagrama de Instancias* y el *Diagrama de Usuarios Instanciados*. Mientras que el primero siempre debe realizase, el segundo sólo se hará si se ha realizado modelado de usuarios, es decir si existen distintos perfiles y modalidades de uso del sistema.

En el *Diagrama de Instancias*, y asumiendo un enfoque descendente, se parte del *Diagrama Estructural* y se crean instancias de nodos simples y compuestos. Por ejemplo, el nodo-tipo "Curso" de la figura 2.4 se instancia en tres nodos que se corresponden con las tres temáticas abordadas en Now-Graduado: Curso.Lengua, Curso.Matemáticas y Curso.Socionaturales. A su vez, dentro de cada una de esas instancias habrá que crear instancias específicas de los nodos simples. Cada instancia tiene un identificador y una etiqueta, formada por la concatenación de la etiqueta procedente del Diagrama Conceptual ("Curso" en el ejemplo anterior) y un descriptor de instancia ("Lengua", "Matemáticas" y "Socionaturales" en el ejemplo), de manera que no se pierda la relación con la entidad conceptual a que pertenece y, al mismo tiempo, se pueda identificar a la instancia.

De forma análoga se procede con el Diagrama de Usuarios, creando instancias de los roles o de los equipos y dando lugar al *Diagrama de Usuarios Instanciado*. Si existe una correspondencia entre las instancias de los nodos y la de los usuarios es conveniente utilizar el mismo identificador para facilitar su equiparación.

Especificación de funciones. En esta actividad se hace una especificación muy detallada del funcionamiento de la aplicación utilizando notaciones próximas al lenguaje de implementación e incluso éste si fuera posible. Con tal fin se crean dos productos para diseñar los dos tipos de funciones del sistema:

– Las *Especificaciones de las Estructuras de Acceso* recogen la descripción detallada de los enlaces virtuales, es decir, de aquellos que se calculan en tiempo de ejecución, y de las herramientas de navegación.
– Las *Especificaciones Detalladas de Funciones* en las que se especifican completamente las funciones que se identificaron durante el Diseño Conceptual y se documentaron en las *Especificaciones Funcionales*.

Especificación de instancias. En esta actividad se proporciona toda la información de las instancias creadas anteriormente. En concreto, se da lugar a cuatro productos: los *Diagramas Internos Detallados de Nodos* y *de Contenidos*, la *Tabla de Acceso* y la *Asignación de usuarios*.

Los *Diagramas Internos Detallados de Nodos y Contenidos* son versiones más concretas de sus homónimos en el Diseño Conceptual en los que las "cajas" o contenidos-tipo se sustituyen por contenidos concretos en forma de referencias a objetos multimedia. Todas las características especificadas en el Diseño Conceptual, tales como atributos, eventos, alineamientos o sincronizaciones se mantienen, si bien puede particularizarse si fuera necesario. En el caso de elementos pertenecientes a un objeto general, y que, por tanto, van a ser heredados por sus hijos, puede decidirse si los hijos heredan el valor del elemento o lo pueden sobrescribir. Por ejemplo, para crear cada una de las páginas de Now-Graduado se creó una plantilla que los diseñadores multimedia rellenaban con los contenidos definitivos. En el caso del nodo "Crucigrama" (véase la figura 2.7) los contenidos mono-instancia ("Corregir" y

"Alfabeto") se rellenaban una única vez para todas las instancias mientras que a los multi-instancia ("Enunciado" y "Cuadrícula") se les asignaba un contenido en cada instancia.

La *Tabla de Acceso* está formada por las instancias concretas de las reglas de acceso definidas con el *Catálogo de Categorías* y las *Reglas de Autorización* durante el Diseño Conceptual, que ahora se actualizan para tener en cuenta los sujetos y los objetos concretos producidos durante esta fase. Esta tabla tiene como filas todos los contenidos y nodos de la aplicación y como columnas sus usuarios (es decir, los roles y equipos concretos). Los valores de cada celda son un permiso de acceso, que puede asumir los valores: acceso denegado, navegación, personalización o edición. En el capítulo 7 se describe en detalle cómo se calculan los contenidos de esta tabla.

Finalmente, se asocian usuarios concretos a los roles concretos, de forma que éstos puedan acceder al sistema y ejercitar sus capacidades de acceso. Esta es la misión de la *Asignación de Usuarios*, producto también opcional. Un mismo usuario puede tener asignado más de un rol. En estos casos el comportamiento por defecto es que el usuario asuma durante el uso del sistema la autorización más permisiva, si bien podrían realizarse implementaciones en las que se obligara a los usuarios a decidir qué rol van a emplear al iniciar cada sesión.

Diseño de la presentación. La última actividad de esta fase es determinar la apariencia de los nodos y contenidos, generando las *Especificaciones de Presentación*. En ellas se indican características de presentación del tipo color del fondo, color del texto, color de los enlaces, volumen del sonido, etc., para cada nodo y contenido concreto. Estas especificaciones dependen obviamente del tipo de objeto involucrado. Así por ejemplo, para un nodo se puede indicar aspectos tales como el color o la imagen que se utilizará en el fondo, el tipo de ventana (emergente, con barras de desplazamiento o de tamaño fijo, etc.) o la duración del nodo. Con respecto a los contenidos se tienen aún más variantes, pues hay que tener que distintos contenidos multimedia tienen distintos aspectos de presentación (tamaño, color, volumen, duración, etc.).

La separación entre estas características, introducida en el modelo de referencia Amsterdam [22], hace posible que se puedan definir distintas características para el mismo objeto con el fin de garantizar la máxima accesibilidad en cualquier situación. Con tal fin debería darse soporte a la definición de reglas de adaptación que determinaran la mejor presentación dadas las características del usuario (v.g., discapacidades físicas, edad, cultura, etc.), de la plataforma de operación (v.g., hardware y software utilizado) y del entorno de uso (v.g., condiciones medioambientales como la luminosidad, nivel de ruido, etc.).

2.3.3 Evaluación

Puesto que los sistemas hipermedia son esencialmente interactivos [31] es preciso evaluar las soluciones de diseño para comprobar si se ajustan a las necesidades y expectativas de sus usuarios. La evaluación no sólo puede hacerse con sistemas o con prototipos, sino que existen técnicas analíticas que permiten evaluar especificaciones y productos de diseño [8]. En ADM se consideran ambos tipos de evaluaciones en esta fase, cuyas actividades se resumen en la tabla 2.3. El Prototipo no será

desarrollado cuando lo que se vaya a evaluar con un método analítico sea un producto de diseño (por ejemplo, los diagramas del Diseño Conceptual). A continuación se explican ambas actividades.

Desarrollo del prototipo. Esta puede ser una de las últimas fases del ciclo o la primera, dependiendo de si se ha aplicado un enfoque descendente o ascendente, respectivamente. En el primer caso, si se parte del Diseño Detallado es posible que el prototipo se pueda generar semi-automáticamente si se ha automatizado el proceso anterior. En cualquier otro caso, se puede hacer uso de cualquier herramienta de autor (v.g., ToolBook, Macromedia Dreamweaver) si se trata de un prototipo o comenzar a implementar el sistema en el entorno elegido. La práctica más habitual y razonable es adoptar un enfoque progresivo en el que se empiece con prototipos rápidos, incluso prototipos "de usar y tirar" [30], y se vayan añadiendo cada vez más funcionalidades hasta llegar al sistema final. Dada la especificidad de esta actividad no se pueden plantear guías o normas genéricas como se ha hecho en el resto de las tareas presentadas hasta el momento.

Evaluación. Esta actividad está destinada a comprobar la utilidad del sistema que se está desarrollando. Es una oportunidad única para detectar fallos de *usabilidad* y posibles mejoras, pero hay que planificar bien la evaluación para que se obtengan resultados realmente valiosos que demuestren si el sistema realmente ayuda al usuario a conseguir sus metas [29].

Con tal objeto ADM propone iniciar la evaluación elaborando un *Documento de Evaluación* con el que se prepare ésta antes de llevarla a cabo, recogiendo las exigencias planteadas desde la comunidad dedicada a la ingeniería de la *usabilidad* que pretende sistematizar el análisis, el diseño y la evaluación de la *usabilidad* [Mayhew, 1999]. Este documento se estructura en siete secciones:

1. Objetivo de la evaluación. En este apartado se enumeran objetivos precisos, claros, no ambiguos y, sobre todo, mensurables.
2. Método de evaluación. Se elige una técnica adecuada, ya sea analítica, empírica, experta o experimental [8] considerando el estado en que se encuentre el desarrollo, los recursos disponibles y la experiencia previa en este tipo de procesos [30].
3. Perfil de los evaluadores. Una vez decidida la técnica hay que seleccionar evaluadores adecuados, teniendo en cuenta los objetivos de la evaluación y el tipo de características que se quieren analizar. Los evaluadores pueden ser usuarios del sistema, ya sean reales o potenciales, o expertos en diferentes dominios relacionados con el sistema.
4. Datos a recoger. Es importante estudiar qué datos cuantitativos y cualitativos se quieren estudiar, para lo cual habría que establecer ciertos criterios. A este efecto ADM plantea los propuestos en [12], que incluyen aspectos como la riqueza, la completitud, la motivación, la autonomía, la competencia, la flexibilidad, la estética, la consistencia, la autoevidencia o la *predecibilidad*. Para estos criterios se establecen una serie de parámetros, por lo que el diseñador debería seleccionar los criterios que le permiten medir sus objetivos y, dentro de estos, los parámetros que sean aplicables.
5. Tareas. Si se quieren estudiar todos los mecanismos de interacción de la aplicación, es conveniente que la interacción del evaluador sea guiada, de manera que no sólo se asegure que se utilizan todas las opciones disponibles sino que también se

puedan detectar errores. Además, hay que asegurarse de que las tareas permiten recopilar todos los datos que se quieren recoger.

6. Mecanismos de registro. Es preciso pensar en los artefactos a través de los cuales se va a recopilar información, ya sean cuestionarios, entrevistas, grabaciones o software de registro, puesto que hay que prepararlos con antelación.
7. Planificación. Recogerá el plan que guiará la evaluación, incluyéndose personal involucrado e hitos importantes, desde la preparación hasta la realización de la evaluación y el análisis de los resultados.

Una vez se haya realizado la evaluación, es necesario analizar los datos obtenidos y extraer conclusiones que se reflejarán en el *Informe de Conclusiones*, en el que se podrán proponer mejoras que afecten a cualquiera de los productos desarrollados en cualquiera de las fases del desarrollo.

El proceso a seguir durante la evaluación se resume en los siguientes pasos:

1. Preparación de la evaluación. Se elabora cuidadosamente el *Documento de Evaluación*. Independientemente de que lo que se evalúe sea un prototipo, un sistema o una especificación (es decir, alguno de los productos generados en las fases previas) es preciso planificar bien esta actividad con objeto de no perder la oportunidad de obtener datos útiles.
2. Realización de la evaluación. Al realizar la evaluación hay que recrear un entorno real. Además, los evaluadores deben ser conscientes de qué se está evaluando y por qué. Las evaluaciones pueden hacerse de forma individual o en grupo. Si se prevé una evaluación larga, se dividirá en varias sesiones para no cansar a los evaluadores, ya que la fatiga puede dar lugar a una falta de interés que falsee los resultados de la evaluación. En algunas ocasiones llevar a cabo una sesión de prueba dentro del propio equipo de desarrollo ayuda a depurar el proceso de evaluación.
3. Análisis de los resultados. Los resultados deben servir para extraer una serie de conclusiones en términos de deficiencias detectadas y mejoras propuestas que permitan revisar el producto para incrementar su utilidad. De este estudio deben salir una serie de propuestas concretas que podrán afectar a productos del Diseño Conceptual, del Diseño Detallado o al Prototipo.

Así pues esta fase sienta las bases de un desarrollo progresivo y centrado en el usuario, realimentando con sus conclusiones al resto de las fases.

2.4 La herramienta AriadneTool

AriadneTool [27] es un entorno destinado al desarrollo de sistemas hipermedia y web que está basado en el método ADM. La versión actual cubre todo el Diseño Conceptual y parte del Diseño Detallado puesto que gran parte de las actividades de esta fase supondrían el desarrollo de una herramienta de automatización completa, en la que se incluyeran por ejemplo herramientas de edición multimedia, labor excesivamente compleja y que se considera más objeto de una labor empresarial que de investigación. AriadneTool también facilita la generación rápida de prototipos en HTML, SML, SMIL y RDF, así como la generación de documentación sobre los

proyectos desarrollados. Además, la herramienta incorpora ontologías para chequear la completitud, consistencia y corrección de los productos generados.

La herramienta se está desarrollando siguiendo un proceso centrado en el usuario, de tal manera que todos los años se evalúa empíricamente en un curso sobre diseño hipermedia para recoger datos sobre su utilidad y su *usabilidad*. El empleo de Java para la implementación, en concreto JDK1.4, proporciona independencia de la plataforma de utilización y facilita la interconexión con otras aplicaciones basadas en tecnología Java.

2.4.1 Arquitectura de AriadneTool

La arquitectura de AriadneTool es la que se muestra en la figura 2.8.

La interfaz permite a los desarrolladores diseñar su sistema generando productos de ADM (véase la tabla 2.3) mediante un estilo de interacción de manipulación directa de objetos. Los productos y los objetos tienen ventanas de propiedades en las que añadir información sobre los mismos, información que se utilizará al generar la documentación del proyecto. Cada diseño se corresponde con un proyecto y se pueden incluir referencias a otros proyectos hechos con AriadneTool a través de las herramientas externas del Diagrama Estructural o del Diagrama de Navegación.

Los repositorios de datos almacenan información sobre los proyectos desarrollados con AriadneTool. El repositorio dinámico contiene todos los elementos del proyecto que está abierto en forma de objetos Java, de forma que se agilice el acceso. En el repositorio permanente los proyectos se convierten en ficheros XML conformes a las DTDs *Entity-Elements DTD* y *ADM Products DTD*. La traducción de objetos java a XML y viceversa se hace mediante JAXB1.1.

El almacenaje de meta-datos contiene las primitivas de modelado de ADM, tanto las entidades básicas (es decir, los elementos de Labyrinth recogidos en las tablas 2.1 y 2.2) como los modelos de diseño (es decir, los productos de ADM resumidos en la tabla 2.3). Para ello se han definido dos DTDs, *Entity-Elements DTD* y *ADM Products DTD* respectivamente.

El modulo de validación comprueba la completitud, corrección y consistencia de los modelos de diseño, notificado al desarrollador cualquier error o sugerencia. Este proceso se lleva a cabo a través de ontologías, tal y como se muestra en la siguiente sección, con las que se describen los modelos de forma completa, formal, procesable por la máquina y comprensible para los desarrolladores.

Finalmente, el generador de prototipos crea automáticamente un prototipo de la aplicación. Puesto que los productos de ADM se han especificado de una forma independiente de la plataforma de operación, es posible generar prototipos en distintos lenguajes, tales como HTML, XML, SMIL y RFD/RDS.

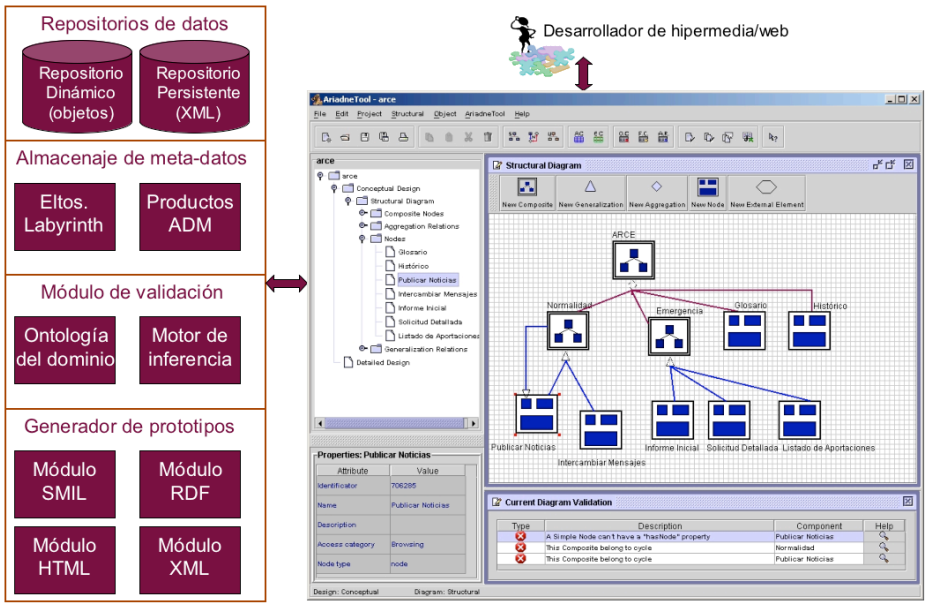


Fig. 2.8.: Arquitectura de AriadneTool, herramienta de automatización de ADM.

2.4.2 Soporte de AriadneTool al modelado con ADM

Con la versión actual de AriadneTool se pueden desarrollar, validar y documentar todos los productos de la fase de Diseño Conceptual y del Diseño Detallado. Cada desarrollo se corresponde con un proyecto en AriadneTool, que englobará tanto los productos de ADM como la documentación asociada a los mismos. La interfaz de la herramienta consta de cuatro áreas:

- El Navegador del Proyecto que permite acceder a todos los productos desarrollados en el proyecto mediante una estructura de árbol. En la zona superior izquierda de la pantalla de la figura 2.8 se puede apreciar dicho navegador parcialmente desplegado. En la figura el árbol comienza con el proyecto e incluye diversas carpetas con los productos desarrollados hasta el momento, estando resaltado el elemento actualmente activo (el nodo "Publicar noticias").
- El Panel de Propiedades muestra los atributos del diagrama o elemento seleccionado (véanse las propiedades de "Publicar noticias" en la zona inferior izquierda de la pantalla de la figura 2.8).
- El Panel de Edición, donde el desarrollador puede generar los productos de ADM empleando manipulación directa como estilo de interacción. En la zona superior derecha de la pantalla de la figura 2.8 se muestra un Diagrama Estructural del Diseño Conceptual.
- El Panel de Validación en el que se muestran los mensajes de error y advertencia generados por el proceso de validación. Por ejemplo, en la zona inferior derecha de

la pantalla de la figura 2.8 aparecen tres errores al haberse detectado una generalización que se inicia en un nodo simple y un ciclo en el grafo de nodos.

Otras características relevantes de AriadneTool son los menús, barras de herramientas y ayuda contextuales, que ofrecen ayuda inmediata al desarrollador sin abandonar el contexto en el que está trabajando.

2.4.3. Validación automática

ADM incluye una serie de reglas de validación de productos (intra-reglas) y entre productos (inter-reglas) destinadas a analizar la completitud, consistencia e integridad de los productos de diseño generados en cada fase del diseño.

- Las intra-reglas comprueban el cumplimiento de una serie de reglas sintácticas establecidas en ADM y que determinan cuándo se considera que un producto ADM está bien formado. Un ejemplo de regla de este tipo es la que establece que las relaciones de agregación o generalización tienen como origen un nodo compuesto.
- Las inter-reglas comprueban que todos los productos son completos y están correctamente definidos, teniendo en cuenta las interrelaciones existentes entre productos. Por ejemplo, se verifica que un enlace definido en el Diagrama de Navegación tenga un ancla en el Diagrama Interno del nodo del que parte.

Para llevar a cabo la validación, el meta-modelo subyacente a ADM, es decir, la estructura de sus productos y sus relaciones semánticas, se ha expresado haciendo uso de una ontología. La figura 2.9 muestra parte de la expresión ontológica del meta-modelo, en concreto la que recoge las relaciones estructurales y semánticas entre elementos de Labyrinth.

Cuando se crea un determinado producto con AriadneTool, éste también se expresa en términos de la ontología, comprobando si los axiomas que representan reglas de modelado se han respetado. En la figura 2.10 puede verse la reescritura de un Diagrama Estructural en RDF.

Esta validación puede realizarse para un único producto, entre productos o de todo el diseño, opción esta última que incluye la validación interna de cada un de los productos realizados hasta el momento y la inter-validación de productos. Como resultado de este proceso, el usuario recibe una serie de mensajes de error o de advertencia (véase el Panel de Validación de la figura 2.8).

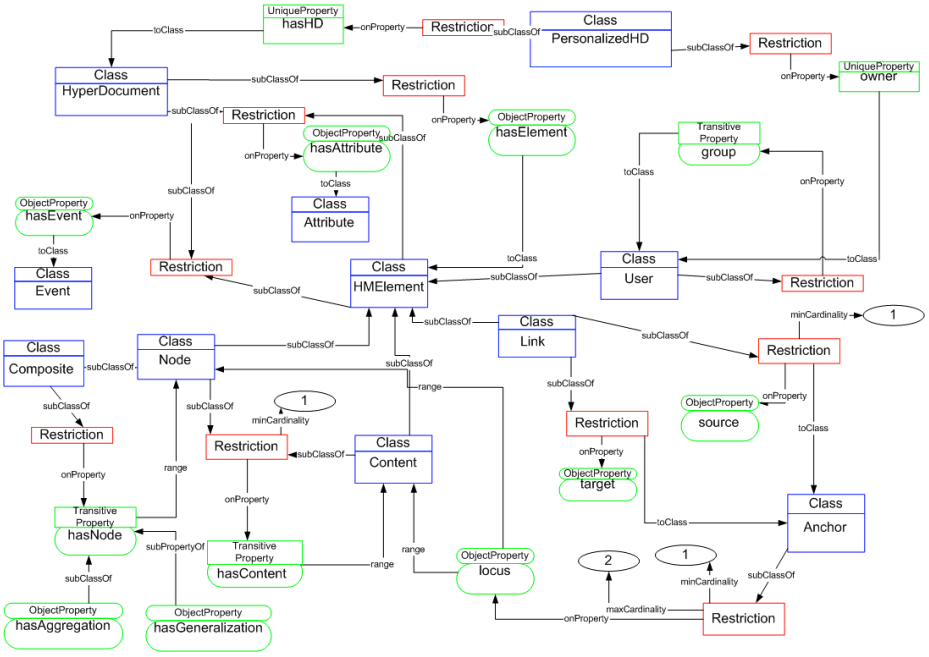


Fig. 2.9.: Fragmento de la expresión ontológica del meta-modelo de ADM.

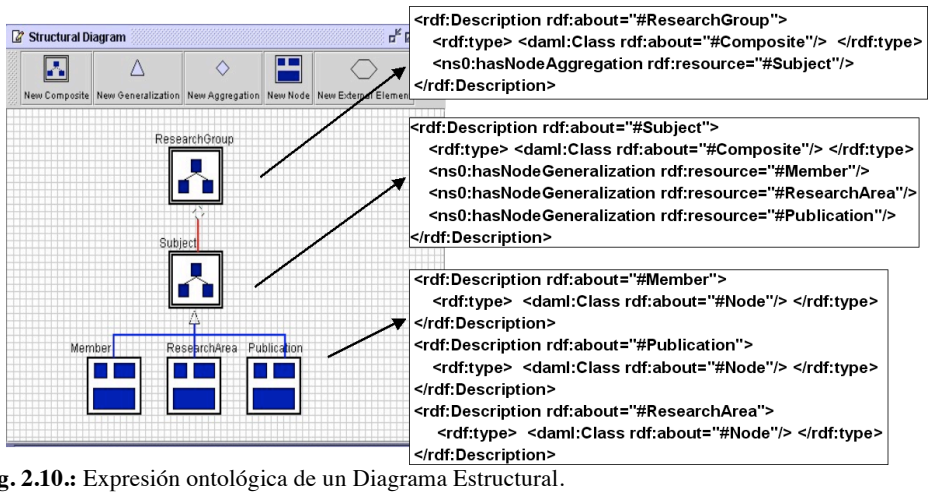


Fig. 2.10.: Expresión ontológica de un Diagrama Estructural.

2.5 ADM y AriadneTool: lecciones aprendidas

Tras haber descrito el método de desarrollo y haber revisado las principales características de la herramienta de automatización esta sección está orientada a resaltar las principales ventajas que supone la aplicación de ADM en el proceso de

desarrollo de sistemas hipermedia y web. En primer lugar se describen algunas experiencias reales de aplicación de ADM, haciendo especial hincapié en las aportaciones que el método ha supuesto, para, a continuación, analizar en qué tipos de desarrollos resulta de especial utilidad la aplicación de ADM.

2.5.1 Experiencias en el desarrollo de aplicaciones reales

ADM se ha empleado para el desarrollo de sistemas hipermedia con muy distintos requisitos e implementados en plataformas absolutamente diferentes, con objeto de analizar su validez práctica y para llevar a cabo un desarrollo iterativo del propio método. También se ha evaluado empleándolo en diversos cursos de diseño de hipermedia [1, 13], en los que los alumnos modelaron sus propias aplicaciones implementadas en diversas plataformas. Estas evaluaciones también han contribuido a la continua mejora de ADM.

Como ya se comentó en la sección 2.3, la primera versión de ADM se empleó para el desarrollo de Now-Graduado. Esta experiencia permitió profundizar en el diseño multimedia, buscando herramientas de modelado que favorecieran la expresión de presentaciones muy dinámicas. Los Diagramas Internos, con su doble visión espacial y temporal y con los alineamientos y las sincronizaciones, se mostraron como un producto fundamental para conseguir un diseño completo y para llevar a cabo una validación con los usuarios antes de iniciar la implementación. Además, la gran variedad de ejercicios interactivos incluidos en Now-Graduado llevó a estudiar el diseño de la interacción y el uso de eventos para añadir aspectos dinámicos al modelo [18].

A partir de aquí se elaboró un método más sistemático y completo, llegándose al estado en que se ha presentado en este capítulo. Con esta versión se han desarrollado sistemas como TurismoYviajes.com, una revista electrónica que podía ser modificada vía web [17], Courba [5], una plataforma web para crear documentación personalizada para cursos o ARCE [3] un sitio web para la gestión de la ayuda internacional en situaciones de desastre que se está desarrollando en el marco de un Convenio de Colaboración suscrito entre la Universidad Carlos III de Madrid y la Dirección General de Protección Civil y Emergencias del Ministerio del Interior. Aunque este último proyecto se describe con mayor detalle en el capítulo 7, a continuación se analizan algunas de las contribuciones que el uso de ADM está suponiendo.

Desde el inicio del proyecto ARCE se evidenció la necesidad de contar con un método bien definido dada la envergadura del mismo, las características del desarrollo y del propio equipo de trabajo, al que se incorporaban de forma muy activa técnicos de protección civil sin conocimientos de aspectos tecnológicos. Por todo ello, se decidió utilizar ADM durante el desarrollo, mostrándose ésta como una decisión muy acertada.

En primer lugar, la fase de Diseño Conceptual fue sumamente útil puesto que hizo posible elaborar una serie de productos abstractos con los que discutir las características del sistema sin tener que recurrir al prototipo, puesto que el uso exclusivo de prototipos desvía frecuentemente la atención hacia aspectos demasiado concretos (organización de las ventanas en la pantalla, tipos de letra o colores)

mientras que se obvian otros, como la forma en que estaba estructurada la información o las distintas modalidades de acceso, de más difícil solución una vez que se ha implementado el sistema. Por ejemplo, el Diagrama de Usuarios, un producto clave en este proyecto junto con el modelado de la política de acceso, tuvo que ser refinado en profundidad puesto que tenía que servir para los 21 países que forman parte de la Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil, que es la destinataria de este ARCE. Este producto representa los tipos de usuarios que existen utilizando una notación sencilla, de manera que se ha podido implicar a los representantes de la asociación para comprobar que se han considerado todos los casos posibles.

En la fase de Diseño Detallado, la mayor parte de las instancias de los nodos y contenidos se especificaron de forma procedimental puesto que se generan dinámicamente. Además, se establecieron las relaciones temporales y espaciales entre los contenidos de la ayuda multimedia para crear una presentación armónica y estética. Las Especificaciones de Presentación se utilizaron para dar soporte a la presentación multilingüe de forma que se pudiera seleccionar el idioma de la interfaz utilizando un atributo del usuario: el país al que representa.

Pero la que sin duda es la fase más importante del desarrollo es la Evaluación. En ella se ha podido contar con potenciales usuarios para analizar las decisiones de diseño, de tal manera que la construcción de este sistema se está haciendo de forma absolutamente centrada en el usuario. Se han evaluado tanto los productos del Diseño Conceptual y Detallado como los distintos prototipos que se han ido construyendo. Los prototipos se han desarrollado sobre un servidor web Apache para el que se han implementado una serie de módulos con Zope, un servidor de aplicaciones que genera páginas HTML dinámicas partiendo de información que se encuentra almacenada en una base de datos relacional PostgreSQL. La primera evaluación empírica realizada con una versión del sistema completamente operativa se realizó en Febrero de 2002 aprovechando una reunión de la Asociación en Cartagena de Indias (Colombia). Los resultados, que se pueden consultar en [3], fueron utilizados para mejorar tanto el diseño como la implementación incluyendo nuevas funcionalidades. Desde entonces se han realizado continuos ejercicios de simulación de forma distribuida que están permitiendo ampliar los servicios ofrecidos por ARCE y, sobre todo, involucrar a todos los miembros de la Asociación en este proyecto de una forma muy activa.

2.5.2 Ámbito de aplicación

El uso de un método como ADM que es sistemático, aunque flexible, es más que recomendable en todos aquellos proyectos que sean de gran envergadura, tanto por la complejidad y tamaño del sistema a crear como por el tiempo de vida que se espera que tenga el mismo, puesto que el método lleva a analizar y a documentar las características estructurales, de navegación, de presentación, de interacción, de funcionamiento y de acceso, desde distintas perspectivas. Todos los productos generados pueden ser evaluados con los usuarios o destinatarios, de manera que pueden detectarse errores o mejoras antes de implementar el sistema. De hecho, si los resultados de la etapa de Diseño Conceptual se evalúan analíticamente con los destinatarios pueden verificarse algunos requisitos sin haber generado el prototipo,

aunque para comprobar la *usabilidad* del producto final siempre habrá que recurrir a evaluaciones empíricas del sistema o de prototipos.

Además, cuando el equipo de desarrollo es multidisciplinar, como en los ejemplos que se comentan en la sección 2.5.1, ADM ayuda a disciplinar a los distintos miembros, al contar con un marco bien definido en el que se indican los productos que cada uno debe generar.

De forma genérica, y a tenor de las experiencias de aplicación del método en casos reales, los mejores resultados se obtienen en proyectos que:

- Incluyan presentaciones multimedia complejas y con muchas posibilidades de interacción, de manera que se pueda aprovechar el potencial expresivo de las definiciones procedimentales mediante eventos y de las relaciones espacio-temporales entre contenidos.
- Conlleven el acceso de distintos tipos de usuarios con distintas capacidades de visualizar y manipular la información, de tal forma que los desarrolladores se puedan beneficiar de poder integrar el modelado de usuarios y de reglas de acceso con modelado del resto de los requisitos del sistema.
- No tengan todos los requisitos establecidos de antemano, por lo que hay que aplicar un método flexible, que facilite tanto el diseño descendente como el ascendente, y que ofrezca un proceso iterativo y progresivo.
- La *usabilidad* del sistema sea un factor crítico y sea preciso adoptar un método centrado en el usuario que haga de la evaluación un proceso fundamental y a realizar a lo largo de todo el desarrollo.
- Haya gestionar cantidades ingentes de información aunque también existan entidades abstractas (tales como, el país, la emergencia, los usuarios nacionales o los supranacionales en ARCE), de forma que un diseño con varios niveles de abstracción sea de utilidad.
- El equipo de desarrollo sea multidisciplinar y, en consecuencia, se requieran productos fáciles de entender por parte de personas sin conocimientos técnicos.
- Gran parte de los nodos y contenidos se generen dinámicamente, por lo que sea preciso poder realizar especificaciones procedimentales.
- Se considere la adaptación de los contenidos a las características del usuario o de la plataforma de uso, de manera que se pueda hacer uso de la separación entre contenido y características de presentación de ADM.

2.6 Conclusiones

La principal conclusión que se extrae de este trabajo es la necesidad ineludible de contar con métodos bien definidos para desarrollar sistemas hipermedia y web de calidad. El uso práctico del método aquí presentado, ADM, en proyectos de desarrollo complejo, tanto por las características del sistema como por las del propio equipo de trabajo, se ha mostrado muy beneficioso pues no sólo ha ayudado a disciplinar al equipo, sino que también le ha proporcionado un vocabulario común como base para el entendimiento, favoreciendo así la participación activa de todos los miembros, incluidos los potenciales usuarios.

Otra conclusión remarcable es que es muy importante que los métodos recojan la experiencia y el conocimiento acumulado en el dominio de la hipermedia. No basta con utilizar un método con el que hacer una especificación formal o semi-formal, como puede ser un diagrama de clases UML, sino que es preciso que el método se sustente en un concepto concreto de hipermedia, de tal forma que ofrezca al diseñador construcciones útiles que, probablemente, desconozca, como por ejemplo, los enlaces n-arios, la separación entre nodos, contenidos y enlaces o la definición de restricciones temporales y espaciales.

Las contribuciones concretas de ADM con respecto a otros métodos pueden resumirse en los siguientes puntos:

- Ofrece productos para especificar todas las características del sistema, incluyendo la estructura de navegación, las características de la presentación, la estructura lógica, los comportamientos, las reglas de acceso y el funcionamiento de la aplicación.
- Incluye mecanismos explícitos para modelar la estructura de usuarios, muy útil no sólo para especificar reglas de acceso sino también para conocer los posibles perfiles de acceso y sus necesidades.
- Ofrece un método completo, integrador e independiente de plataforma en el que se especifican todas las fases y actividades a realizar.
- Hace posible un diseño ascendente y descendente que se ajuste a las necesidades del desarrollo.
- Hace de la evaluación el centro del desarrollo para incitar a una participación activa y temprana de los usuarios.
- Incluye reglas de validación de productos y entre distintos productos para asegurar la completitud, consistencia e integridad de las especificaciones generadas con este método.

Sin embargo, también se es consciente de una serie de aspectos que ADM no tiene actualmente en cuenta y que, como ya se ha comentado, habría que incorporar. Entre ellos, se consideran prioritarios los siguientes:

- Incorporación de patrones de diseño para fomentar la reutilización de componentes de diseño o de implementación [25].
- Dar soporte al modelado para la web semántica [26].
- Mejorar el proceso de evaluación incluyendo métricas para los distintos parámetros y criterios propuestos [12].

Agradecimientos

Este trabajo se ha realizado en el marco de los proyectos “Ariadne”, financiado parcialmente por el Ministerio de Ciencia y Tecnología con referencia TIC2000-0402, y “Sistema de Ayuda para la Construcción de Aplicaciones Hipermedia/Web Basado en el Uso de Patrones de Diseño” financiado por la Comunidad Autónoma de Madrid y el Fondo Social Europeo con referencia 07T/0024/2003.

2.7 Referencias

[1] Aedo, I. y Díaz, P. Applying software engineering methods for hypermedia systems". Proceedings of ACM ITICSE 2001. Canterbury (Reino Unido). Junio. 5-8. 2001.

[2] Aedo, I., Díaz. P., Panetsos, F., Carmona, M. Ortega, S. y Huete, E. A hypermedia tool for teaching primary school concepts to adults. Proceedings of IFIP WG 3.3 Conference "Human Computer Interacion and Educational Tools", Sozopol (Bulgaria). 180–188. 1997.

[3] Aedo, I., Díaz, P., Fernández, C. y Castro, J. Supporting efficient multinational disaster response through a web-based system. E-Gov Conference. Aix-en-Provence (Francia). Septiembre 2-6. R. Traunmüller y K. Lenk (Eds). LNCS 2456, 215-222. 2002.

[4] Aedo, I., Díaz, P. y Montero, S. A methodological approach for hypermedia security modeling. Information and Software Technology, 45(5). 229-239. 2003.

[5] Aedo, I., Montero, S. y Díaz, P. Supporting personalization in a web-based course through the definition of role-based access policies. Interactive Educational Multimedia, 4 (Abril). 40-52. 2002.

[6] Allen, J.F. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11). 832-843. 1983.

[7] Baresi, L., Morasca, S. y Paolini, P. An empirical study on the Design Effort of Web Applications. Proceedings of 3rd International Conference on Web Information Systems Engineering (WISE'02). IEEE Computer Society. 345-354. 2002.

[8] Benyon, D., Davies, G., Keller, L. y Rogers, Y. A Guide to Usability- Usability Now!". Milton Keynes: The Open University. 1990.

[9] Booch, G., Jacobson, I. y Rumbaugh, J. The Unified Modeling Language. Addison-Wesley. 1998.

[10] Brusilovsky, P., Kobsa, A. y Vassileva, J. Adaptive Hypertext and Hypermedia. Kluwer Academic Publishers. 1998.

[11] DeRose, S. y Durand, D. Making Hypermedia Work: A User's Guide to HyTime. Kluwer Academic Publishers. 1994.

[12] Díaz, P. Usability of Hypermedia Educational e-Books. D-Lib Magazine. 9(3). 2003.

[13] Díaz, P. y Aedo, I. A Hypermedia Course in a Spanish Informatics Engineering. ACM SIGCSE Bulletin. 29 (4). 58-61. 1997.

[14] Díaz, P., Aedo, I. y Montero, S. Ariadne, a development method for hypermedia. Dexa 2001, Munich, 4-6 de septiembre. LNCS 2113 ,764–774. 2001

[15] Díaz, P., Aedo, I. y Montero, S. Modelling Hypermedia and Web applications: the Ariadne Development Method. Information Systems. In press.

[16] Díaz, P., Aedo I. y Panetsos, F. Labyrinth, an abstract model for hypermedia applications. description of its static components. Information Systems, 22(8). 447–464. 1997.

[17] Díaz, P., Aedo, I. y Panetsos, F. A methodological framework for the conceptual design of hypermedia systems. Proc. of the fifth conference on Hypertext and Hypermedia: products, tools and methods (H2PTM 99), París, septiembre. 213–228. 1999.

[18] Díaz, P., Aedo, I. y Panetsos, F. Modelling the dynamic behaviour of hypermedia applications. IEEE Transactions on Software Engineering, 27(6). 550–572. 2001.

[19] Díaz, P., Aedo, I. y Panetsos, F. Modelling security policies in hypermedia and web-based applications. En Web Engineering: Managing diversity and complexity of web application development. Murugesan, S. y Deshpande, Y. Eds. , volume LNCS 2016, 90–104. 2001.

[20] Egenhofer, M. y Franzosa, R. Point-set Topological Spatial Relations. International Journal of Geographic Information Systems, 55(2). 160-174. 1991.

[21] Garg, P. K. Abstraction mechanisms in hypertext. Communications of the ACM. 31 (7). 862-870. 1988.

[22] Hardman, L. Bulterman, D. y Van Rossum, G. The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. Communications of the ACM, 37(2). 50-62. 1994.
[23] Kropp, B. y Gallaher, M. P. Access to cost savings: Role-based access control systems can save organizations time and money. Information Security Maganize (Abril). http://www.infosecuritymag.com/articles/april01/cover.shtml#case_study. 2001.
[24] Mayhew, D.J. The usability engineeering lifecycle: a practitioner's handbook for user interface design. Morgan Kaufmann. 1999.
[25] Montero, S., Díaz, P. y Aedo, I. Formalization of Web Design Patterns Using Ontologies. AWIC 2003. LNCS 2663 179-188. 2003.
[26] Montero, S., Díaz, P., Aedo, I. y Dodero, J.M. Toward hypermedia design methods for the semantic web. DEXA International Workshop on Knowledge, Ontology, Metadata and Meaning Matters. IEEE Press. 762-767. 2003.
[27] Montero, S., Díaz. P., Dodero, J.M. y Aedo, I. AriadneTool: A Design Toolkit for Hypermedia Applications. JoDI 5,2, Special issue of on Information Design Models and Processes. 2004.
[28] Nanard, J. y Nanard, M. Hypertext design environments and the hypertext design process. Comm. of the ACM, 38(8).49–56. 1995.
[29] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. y Carey, T.. Human Computer Interaction. Addison-Wesley. 1994.
[30] Preece, J., Rogers, Y. y Sharp, H. Interaction Design: beyond human computer interaction. John Wiley &Sons. 2002.
[31] Rada, R. Hypertext: from Text to Expertext. McGraw-Hill. 1991.
[32] Trigg, R. A Network-Based Approach to Text Handling for the Online Scientific Community. PhD thesis, Univ. of Maryland, 1983.

3 Diseño de Aplicaciones Web con VisualWADE

Jaime Gómez, Antonio Párraga, Oscar Asensi, José Antonio Navarro

Grupo de Investigación en Ingeniería Web
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
03690 - ALICANTE
http://www.dlsi.ua.es/iwad/
e-mail: [jgomez,aparraga, oasensi, jnavarro]@dlsi.ua.es

1 Introducción

Como ya se ha mencionado en los capítulos anteriores, la rápida evolución de Internet en general y de la WWW en particular ha propiciado en los últimos años la investigación intensiva en el campo del modelado conceptual de aplicaciones web. Esta circunstancia ha dado lugar a la aparición de una nueva línea de investigación dentro de la Ingeniería del Software conocida como Ingeniería de la Web (*Web Engineering*). En este contexto se han propuesto distintos métodos, lenguajes, herramientas y patrones de diseño [15, 17] de modelado hipermedia. Algunos de los más relevantes estudiados hasta el momento son HDM [9], HDM-lite [7], WebML [4], OOHDM [19], RMM [12], ADM [1, 10, 14], Strudel [6] u OO-H [11, 3,2]. Estos métodos se centran en su mayor parte en la definición de los aspectos de navegación y presentación respecto a la semántica de los modelos para capturar la problemática del desarrollo en entorno web. Sin embargo, son pocas las propuestas que han intentado aplicar su método en la resolución de casos reales complejos para comprobar la validez y eficacia de sus constructores de modelado. Y mucho menores han sido los intentos (exitosos o no) de construir herramientas de propósito específico para la Ingeniería Web como por ejemplo, WebRatio [21] desarrollada en el marco de un proyecto europeo de V programa marco en el Politécnico de Milano (Italia) bajo la dirección técnica del Prof. Piero Fraternali o ArgoUWE [23] desarrollada en la Luwdig Maximilians University de Munich (Alemania) bajo la dirección técnica de la Dra. Nora Koch o Ariadne [22] desarrollada en la Universidad Carlos III de Madrid (España) bajo la dirección técnica de la Prof. Paloma Díaz (véase capítulo 2)..

Este capítulo describe los fundamentos ingenieriles de VisualWADE una herramienta CASE para el diseño y producción automática de interfaces de usuario de aplicaciones web que está basada en el método OO-H desarrollado en la Universidad de Alicante. La idea que subyace en VisualWADE consiste en que con una combinación adecuada de conceptos simples (constructores de modelado), el diseñador puede modelar y automáticamente generar cualquier tipo de sistema basado en web, desde un portal o intranet de una compañía hasta un sitio web seguro de comercio electrónico. VisualWADE es una aproximación dirigida por el modelo (*model-driven*) que hace énfasis en el análisis de requisitos, el diseño de alto nivel, y

el prototipado rápido. De esta forma una aplicación evoluciona de forma suave desde el primer prototipo al producto final y su mantenimiento es consecuencia natural de su desarrollo. Conforme aparecen nuevos requisitos o cambios en los actuales, simplemente hace falta revisar el modelo conceptual, establecer los cambios y regenerar la implementación.

VisualWADE explota un conjunto de conceptos bien conocidos para capturar la complejidad de las aplicaciones web reales, el método subyacente OO-H (*object oriented hypermedia*) proporciona constructores de modelado específicos para modelar espacios de navegación basándose en una notación compatible con UML. La especificación capturada se compila haciendo uso de técnicas de generación basadas en modelos y produce como resultado una presentación por defecto que puede refinarse desde la propia herramienta para obtener la apariencia final de la interfaz deseada con el consiguiente incremento de productividad de desarrollo.

El capítulo está organizado como sigue: la sección 2 proporciona una breve introducción al método OO-H para familiarizar al lector con la notación de modelado. La sección 3 describe los aspectos básicos de modelado organizados en las tres perspectivas a través de las cuales se modela una aplicación web con VisualWADE (estructura, navegación y presentación). El capítulo 4 presenta aspectos avanzados de modelado describiendo de forma detallada cómo se soporta la consistencia entre modelos, cómo se puede extender la herramienta por terceros mediante tecnología de *plug-ins* y, finalmente, una breve descripción de los compiladores de modelos que producen los artefactos software finales. La sección 5 describe las lecciones aprendidas por el equipo de desarrollo que ha creado VisualWADE, estas lecciones son fruto de la aplicación de VisualWADE en el desarrollo de proyectos reales de Ingeniería de la Web sobre varios dominios de aplicación (gestión tributaria, bancos por Internet y firma electrónica). Finalmente, el capítulo finaliza con unas conclusiones que intentan hacer una reflexión, basada en nuestra experiencia, sobre la evolución y el futuro de los métodos, técnicas y herramientas de Ingeniaría de la Web.

2 Una breve introducción al método OO-H

El método OO-H es un método genérico que está basado en el paradigma orientado a objetos y que ofrece al diseñador una notación y una semántica específica para el desarrollo de interfaces basados en web, su conexión con funciones CRUD (*create, read, update, delete*) sencillas o con módulos de lógica preexistentes (*legacy software*).

OO-H define un conjunto de diagramas, técnicas y herramientas que juntos constituyen una propuesta completa para el modelado de interfaces web. El método incluye:

- Proceso de Diseño

- Diagrama de Acceso Navegacional (DAN)
- Diagrama de Presentación Abstracto (DPA)
- Una herramienta CASE que soporta y automatiza gran parte del proceso de desarrollo.

Con OO-H una aplicación tradicional de escritorio puede ser extendida añadiendo dos nuevas vistas (diagramas) complementarias a las vistas de estructura (diagrama de clases) y comportamiento (diagramas de interacción y de transición de estados). La primera de ellas es el DAN con la cual se puede especificar una vista de navegación, la segunda, el DPA captura conceptos relacionados con los detalles de presentación de la interfaz final.

El DAN enriquece un modelo de dominio constituido por el diagrama de clases y los casos de uso [18], con características de navegación e interacción. También define restricciones sobre la navegación y la información que se muestra por cada clase. Para ello, OO-H hace uso del lenguaje de restricción de objetos proporcionado por UML denominado OCL (*object constraint language*) [16, 20]. De esta forma se consigue obtener un diagrama de navegación suficientemente preciso. Por otro lado, la definición de páginas abstractas independientes de la implementación, está basada en un conjunto de plantillas XML que capturan las propiedades relevantes respecto a los aspectos de apariencia de la interfaz en construcción.

A continuación veremos con un mayor nivel de detalle estos diagramas.

2.2 El diagrama de acceso navegacional

El modelo de navegación se captura a través de uno o más DAN´s. El diseñador deberá construir tantos DAN´s como escenarios de navegación quiera representar. El diagrama está basado en cuatro tipos de constructores de modelado: clases, destinos y enlaces navegacionales, y agrupadores de enlaces. Además, cuando se está definiendo la estructura de navegación es necesario tener en cuenta aspectos relacionados con el comportamiento de la navegación, la selección de población de objetos navegables, el orden en que serán navegados, y la forma en que se visualizarán las poblaciones de objetos navegadas, entre otros. Estas características se capturan según diferentes tipos de patrones de navegación y filtros asociados sobre los enlaces. Los constructores de modelado de OO-H son:

- **Clases de Navegación** (CN): se corresponden con clases del dominio cuya visibilidad de atributos y métodos se restringe de acuerdo a los permisos de acceso de usuario y/o requisitos de navegación.

- **Destinos de Navegación** (DN): agrupan elementos del modelo que trabajan en la cobertura de un requisito navegacional genérico.

- **Enlaces de Navegación** (EN): definen los caminos de navegación que un usuario puede seguir a través del sistema. Pueden tener asociados patrones y/o filtros de navegación para enriquecer la semántica del modelo de navegación. OO-H propone 3 tipos de enlaces:

    - **Ei** (enlace interno) define caminos de navegación entre elementos del modelo.
    - **Er** (enlace punto de entrada) punto de inicio de la navegación por cada destino navegacional.
    - **Es** (Enlace de servicio) muestras los servicios disponibles en la vista de navegación. También capturan la forma en la cual se introducen los parámetros para la invocación del método asociado al servicio y la posibilidad de representar por donde continúa la navegación una vez ejecutado el servicio.

- **Agrupadores de enlaces** (AE): proporcionan al usuario con nuevas formas de acceso a la información. En este caso sirven como un mecanismo de abstracción para agrupar enlaces de navegación.

Como ya hemos comentado anteriormente los **filtros de navegación** se capturan mediante expresiones OCL. OO-H distingue entre dos clases de filtros; filtros en origen (o precondiciones) y filtros en destino. Los primeros capturan restricciones de navegación en origen, es decir, condiciones aplicadas sobre el/los objeto/s origen de un enlace, si estas condiciones no se cumplen se inhibe la navegación. Los segundos, son restricciones que acotan la población de objetos que se visualizará.

Los **patrones de navegación** en OO-H caracterizan la forma en la cual se visualiza la información que se va recorriendo en un camino de navegación, la mayoría de los patrones propuestos por OO-H son bien conocidos puesto que provienen del campo de investigación de la hipermedia. Entre otros se encuentran: indexación, mostrar todos y visita guiada.

Además de los filtros y patrones de navegación, OO-H asocia información complementaria sobre los enlaces para capturar propiedades avanzadas de navegación. Entre éstas tenemos:

- **Visualización** (mostrar en origen | mostrar en destino): cuando un enlace conecta dos clases navegacionales. Con esta propiedad se puede indicar si se desea que la información de ambas clases aparezca en un mismo origen (misma página abstracta) o con un salto de navegación implícito (distinta página abstracta).
- **Interacción de Usuario** (manual | automático): esta propiedad permite que un enlace se active o no de forma automática al alcanzar una determinada

clase de navegación. Algunas veces es útil que el usuario no se vea obligado a seleccionar un enlace para acceder a la información destino.

- **Ámbito de Aplicación** (simple | múltiple | universal): Esta propiedad establece el número de objetos que un determinado enlace abarca en origen cuando es atravesado. Tal origen puede referirse a un objeto simple, a un conjunto de objetos (múltiple) o a todo el conjunto de objetos que se visualizan en la vista actual (universal)

2.3 El diagrama de presentación abstracta

El DPA permite el refinamiento de la estructura de interfaz y de los rasgos de visualización capturados en las etapas previas del modelo. Concretamente, los patrones capturados a nivel de DAN, junto con las características de objetos y atributos, proporcionan la información mínima necesaria para generar de forma automática un DPA por defecto.

En OO-H se adopta una aproximación basada en plantillas [1], [6], [7], [14] para la especificación tanto de la apariencia visual como de la estructura de página de la interfaz hipermedia. El uso de plantillas favorece la reutilización de las experiencias de diseño y facilita la consistencia tanto dentro de los distintos módulos que componen la aplicación como entre distintas aplicaciones.

OO-H define cinco tipos de plantillas, expresadas como documentos XML (*eXtensible Markup Language*) [5], [13]. XML permite al diseñador mantener información acerca del significado semántico de los distintos datos capturados en OO-H. Para ello especifica un marco estandarizado dentro del cual definir las etiquetas necesarias para capturar dicha información. Las páginas resultantes pueden ser directamente visualizadas o traducidas a cualquier otro lenguaje para su publicación. Con esta aproximación, la adición de nuevos tipos de plantillas es muy sencilla.

Las etiquetas y la estructura del documento se han definido mediante un conjunto de DTDs (*Document Type Definition*), uno para cada tipo de plantilla. La DTD especifica el conjunto de reglas que debe cumplir un documento XML para ser considerado válido. Como las páginas abstractas son documentos XML, la DTD especificará las etiquetas que puede contener ese tipo de página, las anidaciones válidas de etiquetas, sus atributos y los valores permitidos. También contiene otro tipo de información, como son instrucciones de proceso, comentarios, declaración de tipo de documento etc. Los tipos de plantilla definidos hasta el momento en OO-H son:

- `tStruct`: estructura la información que debe aparecer en la página materializada.
- `tStyle`: define los rasgos de visualización como son el emplazamiento físico de los elementos, la tipografía o la paleta de color.

- `tForm`: define las estructuras de información que debe introducir el usuario para interactuar con el sistema.
- `tFunction`: captura funcionalidad de cliente mediante funciones definidas
- basándose en el modelo DOM (*Document Object Model*) [5].
- `tWindow`: su uso implica la existencia de dos o mas vistas de información activas en un mismo instante de tiempo.

Se puede obtener mayor información respecto al contenido de las plantillas en [11].

3 VisualWADE: Aspectos Básicos

3.1 Modelado del dominio

El punto de partida para abordar el diseño de una aplicación web es el modelo de dominio representado con un diagrama de clases (ver Figura 1). La notación de este diagrama está basada en UML con lo cual es bastante intuitivo su uso por parte de un diseñador familiarizado con técnicas de análisis y diseño orientado a objetos. La aplicación te asiste en todo el proceso de edición mediante un simple y a la vez potente e intuitivo interfaz gráfico. La creación de clases, atributos, relaciones jerarquías de herencia, etc. se realizan con una simple selección. La disponibilidad en el entorno de botones de *zoom* y vistas generalizadas ayudan a organizar y manejar diagramas complejos compuestos por varias clases.

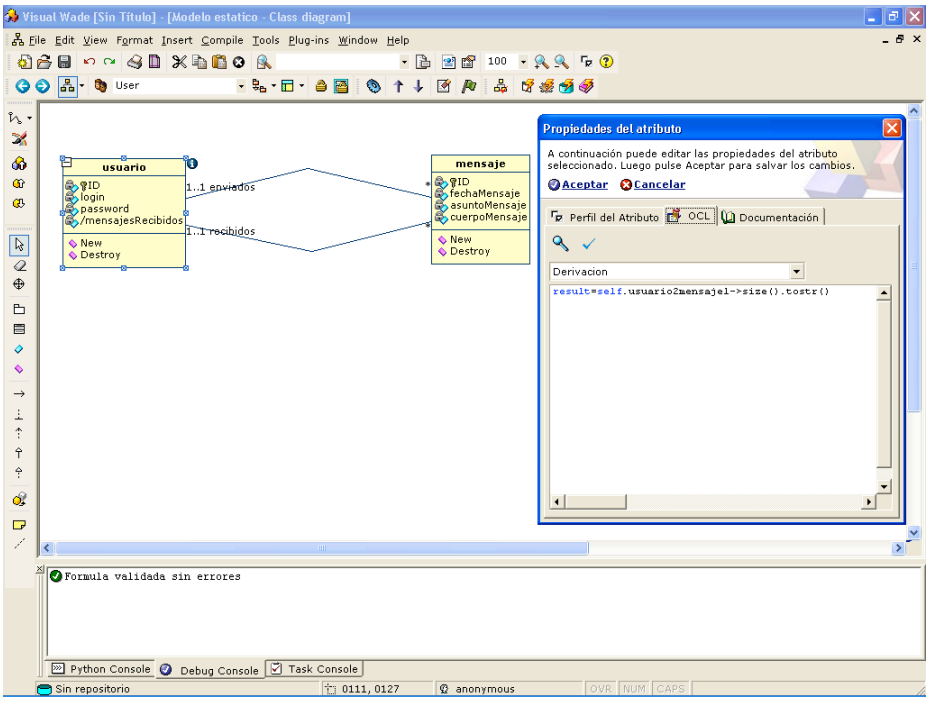


Figura 1: Diagrama de clases en VisualWADE

En la Figura 1 se puede observar el diagrama de clases correspondiente a un sistema de correo electrónico basado en web (`webMail`). En este caso, se especifica la clase `usuario` y la clase `mensaje` y entre ellas relaciones de asociación para capturar la información correspondiente a los mensajes `enviados` y `recibidos` por parte de un usuario.

Además se puede especificar atributos derivados. Para ello, se estereotipa el atributo de interés como `derived` y se le asocia una fórmula OCL que especifique la obtención del valor de ese atributo. Este es el caso del atributo `mensajesRecibidos` de la clase usuario, cuyo valor se obtiene navegando por el rol `usuario2mensaje1`, calculando el numero de instancias de la clase mensaje (aplicación de la función `size()`) y finalmente pasándolo a cadena (aplicación de la función `str()`). Una de las características del entorno es que integra un compilador de OCL que además permite validar sintáctica y semánticamente las fórmulas introducidas.

3.2 Modelado del Espacio de Navegación

Una vista de navegación refleja la forma en la que se accede al contenido de información del modelo de dominio y a los servicios proporcionados por las clases. En la Figura 2 el lector puede observar una vista parcial del diagrama DAN para un usuario del sistema. El punto de entrada es un agrupador de enlaces `principal` que representa una página abstracta de inicio. Desde `principal` se puede navegar a la clase de navegación `usuario` a través del enlace `autenticar`. La activación del enlace `autenticar` requiere que el usuario proporcione información respecto a su nombre de usuario (`dst.login`) y su palabra de paso (`dst.password`) con el objetivo de comprobar si es un usuario válido en el sistema. Dependiendo de si este proceso de validación tiene éxito o no las precondiciones asociadas a los enlaces `LI2` y `LI3` determinan el enlace que se activará y, por tanto, el agrupador de enlace destino (`menu` o `error`). Las funciones OCL `notEmpty()` y `isEmpty()` proporcionan la información necesaria para conocer si un usuario está o no registrado en el sistema. Además desde el agrupador `principal`, y a través del enlace `LI2`, se accede a una clase de navegación `usuario` que ofrece el servicio `nuevoUsuario`, primitiva proporcionada por el constructor de la clase. Finalmente, desde el agrupador `menu` se continua la navegación por el destino navegacional `gestión mensajes` mientras que con el agrupador `error` se vuelve a la página de inicio `principal`.

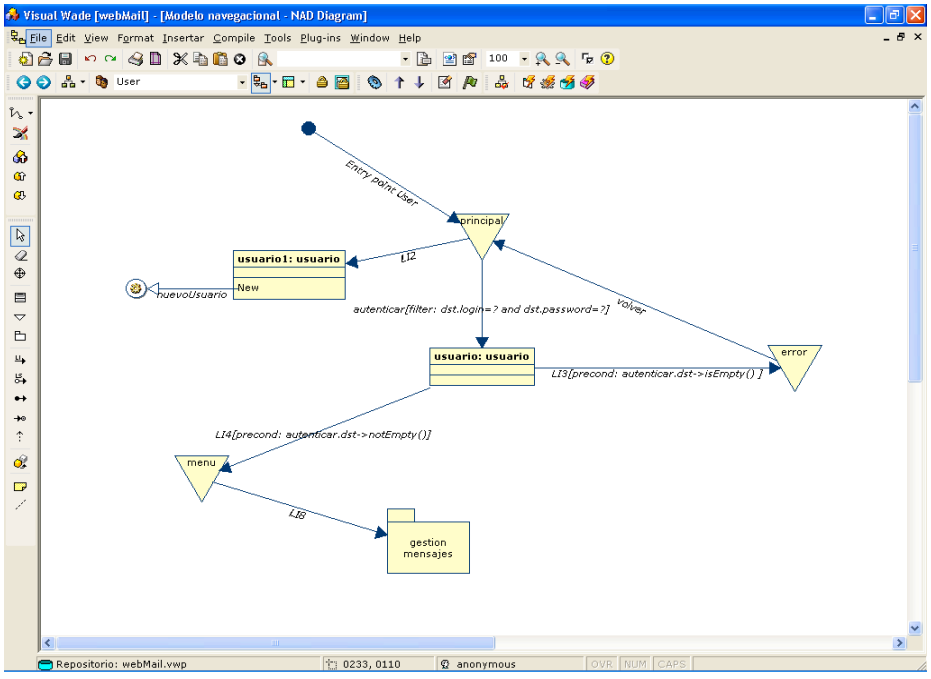


Figura 2: Diagrama de Navegación en VisualWADE

La barra de herramientas de la parte izquierda contiene todos los constructores de modelado que se pueden insertar en un DAN. Haciendo doble clic se accede a las propiedades internas de cada elemento. Los enlaces permiten construir los caminos de navegación indicando si la información se mostrará en la misma o distinta página abstracta así como el conjunto de operaciones que están disponibles para su activación. El entorno visual, además, incluye funciones muy útiles como copiar/pegar, hacer/deshacer y zoom in/zoom out, búsqueda rápida de elementos y finalmente reglas para comprobar la validez del modelo (*model checking*) y mensajes de aviso aspectos que se presentan más adelante.

3.3 Modelado de la Presentación

Una de las características más reconocidas del entorno y que hasta ahora no se ha incluido en ninguna de las herramientas CASE actuales es la posibilidad de compilar el diagrama de navegación, generar el conjunto de páginas abstractas mínimo que cumple con las especificaciones de navegación y ofrecer un diagrama abstracto de presentación para modificar de forma visual la representación final de la interfaz. Los aspectos que se pueden refinar son los correspondientes con las propiedades de estilos, localización, colores, inclusión de componentes finales (imágenes, presentaciones Flash,…) tal y como podría hacerse con cualquier herramienta de autor (ej. frontpage, dreamweaver) pero con la posibilidad de mantener en todo momento la consistencia con las vistas anteriores (diagrama de clases y de navegación). De esta forma se consigue un desarrollo incremental de la aplicación web.

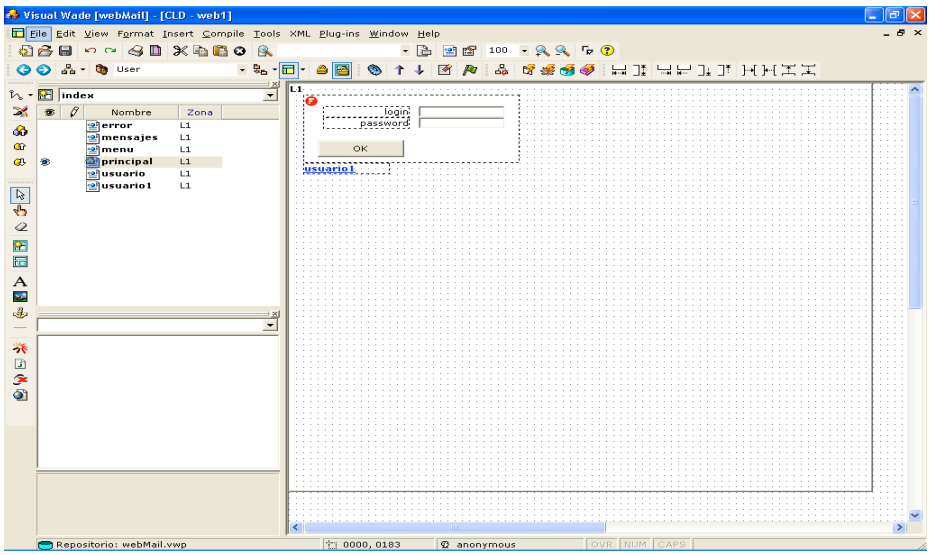


Figura 3: Diagrama de Presentación en VisualWADE

La Figura 3 muestra el diagrama de presentación abstracto resultado de compilar el DAN de la Figura 2. Se observan varias zonas, en la parte derecha tenemos el `visor de páginas` que contiene las páginas abstractas que se han generado ordenadas alfabéticamente (`error`, `mensajes`, `menu`, `principal`, `usuario`, `usuario1`). En la parte derecha tenemos el `área de edición` donde se visualiza el contenido de las páginas abstractas. En este caso se está mostrando la información de la página `principal`. Como resultado del proceso de compilación se ha generado un formulario con los campos `login` y `password` y el botón correspondiente para ejecutar la acción (`OK`). Además se observa el enlace `usuario1` que habilita la navegación para dar de alta un usuario.

El entorno ofrece la posibilidad de animar la interfaz generada con el objetivo de comprobar que efectivamente se está diseñando la solución adecuada. Cuando la herramienta de animación está activa los enlaces y botones de la interfaz resultante se vuelven sensibles a las selecciones de ratón produciéndose en el entorno el salto de navegación correspondiente. Mientras se está animando un modelo es posible seleccionar la herramienta de edición para modificar cualquier propiedad de los elementos de la interfaz. En la Figura 4, se pueden observar estas características.

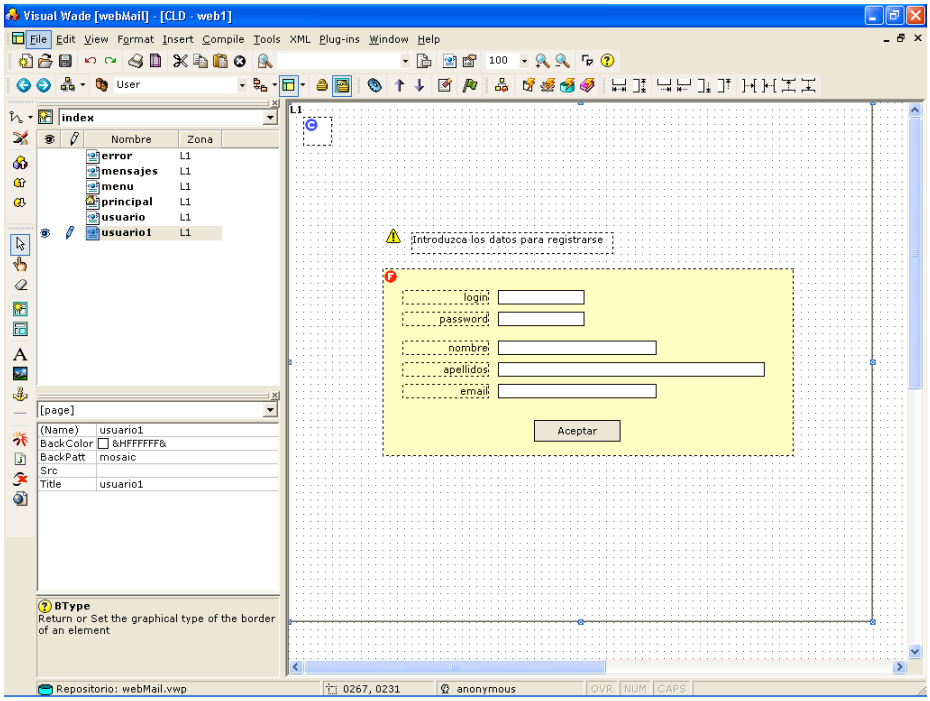


Figura 4: Refinamiento y animación en el DPA

La Figura 4 es el resultado de haber activado la herramienta de animación, haber seleccionado el enlace `usuario1` de la Figura 3 y finalmente haber usado la herramienta de edición para modificar el aspecto de esta página. Se puede observar que entre estos refinamientos se ha incluido un texto de ayuda y se han cambiado las posiciones tamaño y color del formulario de creación de usuarios.

4 VisualWADE: Aspectos avanzados

4.1 Comprobación Automática de Modelos

VisualWADE incluye una potente funcionalidad para comprobar de forma automática la validez del modelo que se está diseñando. De esta forma se verifica tempranamente la especificación capturada con el consiguiente ahorro de tiempo en el proceso de generación de código. El proceso de comprobación se produce a tres niveles:

- Comprobación del modelo (*model checking*): esta función valida la construcción de las vistas de estructura y navegación y presenta los problemas detectados de consistencia así como lo que hay que hacer para arreglarlos.
- Comprobación del mapeado de datos (*mapping checking*): esta función comprueba si los elementos del diagrama de clases están correctamente representados en la base de datos generada debido a cambios en la especificación conceptual o el renombrado de atributos de las clases.
- Comprobación de la presentación (*presentation checking*): Este proceso comprueba si la representación de las páginas abstractas se corresponde con la especificación capturada en el diagrama de navegación proporcionando los avisos correspondientes en el entorno para corregir el error.

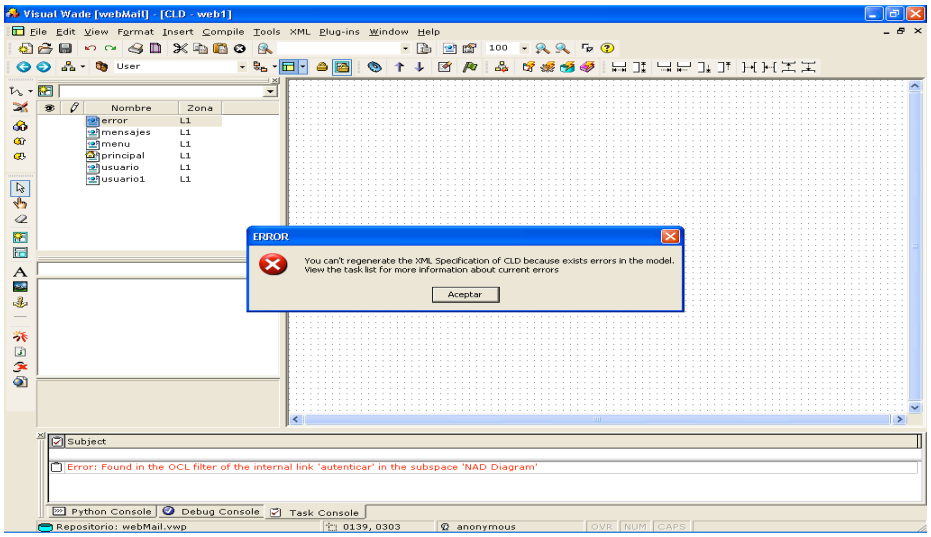


Figura 5: Comprobación del Modelo en VisualWADE

En la Figura 5 se puede observar un aviso del entorno en el proceso de comprobación del modelo. Concretamente, se está informando que no se puede generar el DPA por defecto debido a que se ha encontrado un error en la fórmula OCL asociada al enlace `autenticar` de la vista DAN. Observando con más detalle este error comprobaremos que la fórmula hace referencia a atributos que no han sido declarados en el diagrama de clases.

4.2 *Plug-ins* en VisualWADE

VisualWADE admite la incorporación de *plug-ins* o componentes externos, lo que permite que se pueda ampliar la funcionalidad por parte de terceros.

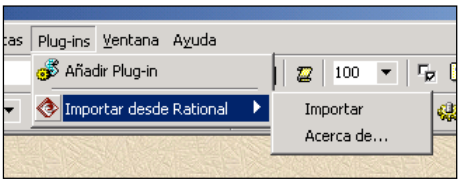


Figura 6: Ejemplo de submenú en un *plug-in*

Estos componentes pueden interactuar con el metamodelo con total libertad y realizar determinadas acciones programadas, tales como generar código, manipular la estructura dinámica del mismo o simplemente realizar alguna tarea a modo de "macro". A nivel interno, un *plug-in* VisualWADE está formado por:

- Definición de interacción con el usuario: Parte de código que define la forma en la que el usuario va a poder ejecutar la funcionalidad del plug-in.
- Funciones Python: Lógica que será invocada como resultado de utilizar el *plug-in*.

La definición de interacción con el usuario contiene información acerca de los menús a añadir a VisualWADE (dentro del menú *Plug-ins*) y las funciones que accionarán los mismos.

```
"""PLUGIN:

    /* Datos del menú: */
   MENUNAME="Importar desde Rational"
   MENUICON="rational.ico"
   MENUITEMS
      MENUITEM="Importar",importRational
      MENUITEM="Acerca de...",aboutImportRational
   FMENUITEMS

   def importRational associated_to cargaPetalFile
   {
         OpenFileBrowser.title='Importar'
         OpenFileBrowser.filter='*.mdl','Ficheros de
rational rose (*.mdl)'
         Launch (OpenFileBrowser)
         Body.WaitMsg="Trasformando fichero petal
(.mdl) en repositorio LES Objects."
         Launch (Body)
   }

def aboutImportRational
   {
       MsgBox.Msg="Plugin para importar ficheros desde
Rational Rose\n\n\tPor Antonio
Párraga\n\tparraga@visualwade.com"
         MsgBox.Title="Acerca de..."
         Launch (MsgBox)
   }

"""

Resto del módulo python...
```

Tabla 1. Estructura de la cabecera de un plug-in

Un mismo *plug-in* puede tener varias funciones y por lo tanto varios submenús. Cada submenú accionará una función del *plug-in*, que a su vez hará dos cosas: pedir

(si es necesario) al usuario información de control y ejecutar un método definido en el código del *plug-in*.

En la Figura 6 se puede observar un menú correspondiente a un *plug-in* para importar modelos de Rational Rose a VisualWADE. Al seleccionar en el menú "Importar" nos saldrá un cuadro de diálogo que nos pedirá la localización del fichero que deseamos importar. Una vez seleccionemos el fichero deseado se llamará automáticamente a la función Python que esté asociada pasándole 2 argumentos: la localización (*path*) del directorio donde está el fichero en formato Rational Rose y una referencia al metamodelo actual. El resto del código de un *plug-in* es código 100% Python. Lo único que diferencia a un *plug-in* VisualWADE de una librería de funciones Python es la información que contiene la cabecera que define los menús y las relaciones de cada opción (ítem) del menú con cada función Python.

Por lo tanto, reaprovechar una librería Python de las miles de librerías existentes en Internet para convertirla en un *plug-in* VisualWADE es tan sencillo como escribir la información de cabecera y asociar las funciones que se van a ejecutar. La información de cabecera se codifica entre comentarios Python comenzando por la palabra reservada PLUGIN y utilizando una sintaxis propietaria que VisualWADE sabe interpretar al importar el *plug-in*. Siguiendo con el ejemplo del importador de ficheros Rational Rose, la estructura de la cabecera del plug-in es similar al que aparece en la tabla 1. En este ejemplo, el código Python deberá contener una función llamada "cargaPetalFile" que es la que estará asociada al menú "Importar".

4.3 Compiladores de modelos

VisualWADE cuenta con tres compiladores de modelos:

- Generador de lógica
- Generador de interfaz
- Generador de Base de Datos.

Cada compilador puede ser invocado de manera independiente para regenerar aquella capa que se desee dejando inalterado el resto. A su vez, cada capa cuenta con una lógica generadora distinta que producirá como resultado el conjunto de ficheros completo que constituye esa capa. Cada generador recibe a su vez una referencia del metamodelo y simplemente se dedica a recorrerlo por completo disparando acciones generadoras por cada elemento que se encuentre a su paso.

4.3.1 Qué se genera

Antes de ver en que se basan los compiladores será interesante ver la estructura de un proyecto generado con VisualWADE:

Capa de Interfaz: Está constituida por un conjunto de páginas que se comunican con la lógica por medio de un objeto (*Object Mediator*) al cual se hacen las peticiones

que tenga cada una programada, tales como invocación de servicios, petición de datos, interrogar acerca de si se cumplen o no determinadas condiciones, etc…

A su vez, las páginas de interfaz pueden estar basadas en plantillas (*templates*), por lo que contendrán una pequeña lógica para rellenar el contenido de las plantillas y visualizar el resultado (nuestra página web http://www.visualwade.com es un claro ejemplo de generación mediante uso de plantillas)

En resumen, las páginas de la capa de interfaz no tienen ninguna lógica de negocio integrada en las mismas y simplemente se dedican a interrogar al *Object Mediator* y a mostrar los resultados.

Capa de lógica: Está constituida por la clase *Object Mediator* y distintos módulos de funciones que serán invocados por el *Object Mediator* por cada petición de la interfaz. De entre los módulos que se generan podemos destacar:

- *Ficheros de consultas*: Contiene todas las consultas a base de datos y la lógica necesaria para rellenar unas estructuras de datos propietarias que sabrá interpretar la capa de interfaz para mostrar los resultados.

- *Fichero de servicios*: Contiene todos los servicios que fueron definidos en el diagrama DAN. Básicamente realizan las operaciones de altas, bajas, modificaciones en la base de datos.

- *Fichero de fórmulas*: Contiene todas las funciones que manipulan datos para transformarlos dinámicamente en otros como resultado de la interpretación de los atributos derivados, las etiquetas DAN, etc…

- *Fichero de resolución de contexto*: Contiene todas las funciones que permiten extraer información de la sesión de usuario tales como las instancias que estuvieron implicadas cuando se ejecutó un determinado filtro, si el usuario navegó o no por un determinado enlace, etc…

- *Fichero de condiciones:* Contiene todas las funciones necesarias para validar entre otras cosas las precondiciones que fueron definidas en el modelo. Este fichero es similar al fichero de fórmulas, solo que el dominio de todas las funciones definidas en este fichero es booleano y simplemente le sirve al *Object Mediator* para comprobar si se cumple o no en cada momento las condiciones definidas por el usuario.

Se cuenta también con un generador de base de datos que genera para distintos motores relacionales permitiendo a su vez que la lógica a generar se pueda definir para uno u otro. Elegir un motor hará que se generen las consultas y servicios con una u otra sintaxis (en ocasiones incluso optimizadas para el motor destino), haciéndoselo transparente al resto de capas.

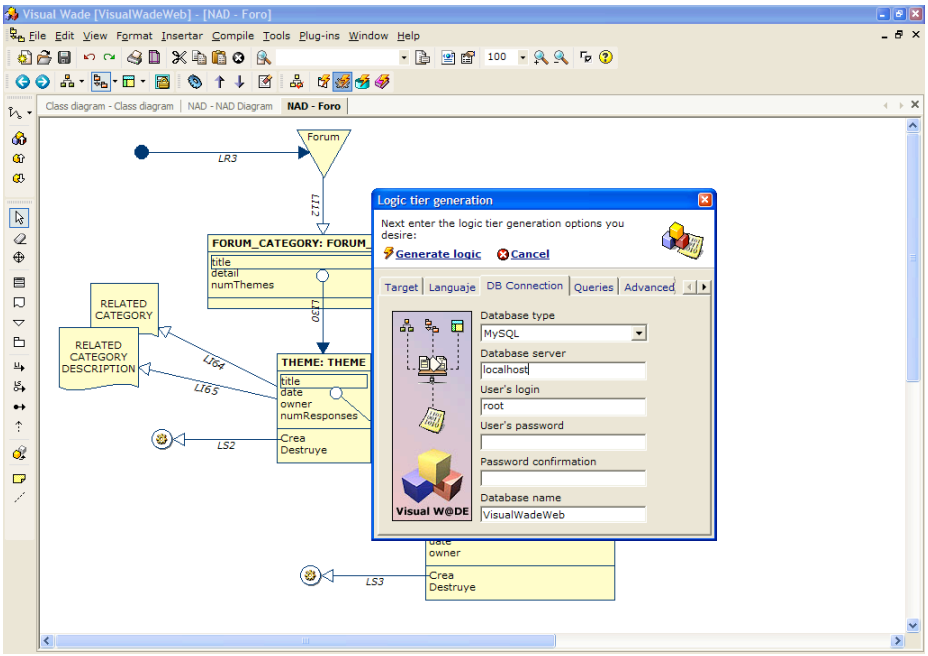


Figura 7: Parámetros de conexión con la base de datos en el generador de lógica

4.3.2 Cómo se genera

Una de las características más interesantes con las que cuenta VisualWADE es la filosofía que utiliza para generar código fuente. Los compiladores de modelos tienen como única misión la de recorrer el metamodelo y disparar acciones generadoras. Sin embargo, dichos compiladores desconocen el lenguaje sobre el que se va a generar código fuente, rellenando únicamente unas estructuras de memoria que representan código fuente sin serlo.

Por otra parte, unos traductores recorren estas estructuras generando el código fuente asociado a las mismas guardándolo en ficheros en disco. A su vez, estos traductores desconocen por completo el modelo (no tienen ninguna referencia al mismo) sobre el que se han basado los compiladores para generar las estructuras.

Tener divida la funcionalidad de generación de esta forma proporciona muchas ventajas, entre las que cabe destacar:

- Se puede incorporar un nuevo lenguaje destino con tan solo programar un nuevo traductor de estructuras de memoria.
- Se puede adaptar un traductor de una a otra versión del mismo lenguaje sin tocar los compiladores.

- Se puede cambiar aspectos técnicos de la generación en ficheros (dónde se van a grabar los ficheros, qué información y cómo la contendrán, etc…) sin manipular los compiladores.
- Se puede cambiar un aspecto en los compiladores de modelo y afectará a todos los lenguajes sobre los que se genera.

Finalmente, también resaltar que con esta filosofía se cumplen las condiciones necesarias para compatibilizar el entorno de desarrollo de VisualWADE según la propuesta MDD/MDA (*model-driven development/model-driven architecture*).

Los compiladores son el gran peso de la generación de VisualWADE: Interpretan las fórmulas OCL para cada contexto conociendo y produciendo de forma distinta para cada caso, analizan exhaustivamente todos los caminos de navegación y determinan cómo y qué generar teniendo en cuenta cómo está montado el diagrama de clases para cada consulta o servicio con el que se encuentren. Los traductores contienen una lógica muy pequeña constituida únicamente por reglas que definen qué generar ante cada estructura de memoria que se vayan encontrando. Todos los traductores son clases que heredan de una clase con métodos genéricos para recorrer las estructuras de memoria, por lo que ni siquiera se tiene que programar en ellos dicha funcionalidad.

En definitiva, esta división permite adecuarse a la evolución de los lenguajes con un mínimo esfuerzo. Se han realizado interesantes experimentos con VisualWADE para generar Java y VisualBasic, pero se ha potenciado sobre todo el traductor de PHP.

A su vez, puesto que los modelos que se generan utilizan bases de datos, el generador de las sentencias SQL también cuenta con sus propias estructuras de memoria que son igualmente traducidas y optimizadas para el motor de base de datos sobre el que se genera. Por ejemplo, en determinadas versiones de MySQL no se permite la utilización de EXISTS o FORALL, mientras que en ORACLE sí. Este aspecto debe ser transparente para el compilador, indicando lo que se desea generar pero no cómo se debe hacer. Será posteriormente el traductor el que se enfrentará a este y muchos otros problemas.

Esto permite entrelazar de una forma muy limpia los distintos lenguajes de código fuente con los distintos motores de base de datos, realizar un fácil mantenimiento de los generadores y, en definitiva, adaptar rápidamente a la evolución del mercado.

Con todas estas funcionalidades presentadas creemos que VisualWADE constituye una apuesta seria como herramienta avanzada para el diseño y generación automática de interfaces de usuario de aplicaciones web. VisualWADE y su notación subyacente (OO-H), han sido el resultado de 4 años intensos de investigación en el campo de la Ingeniería Web en la Universidad de Alicante.

5 VisualWADE: Lecciones aprendidas

5.1 Ámbito de aplicación

VisualWADE se ha utilizado en entornos tan distintos como el Organismo Autónomo de Gestión Tributaria de la Provincia de Alicante (SUMA) [24], la Caja de Ahorros del Mediterráneo (CAM) [25], el Colegio de Ingenieros Técnicos Industriales de la Provincia de Alicante (COITI) [26], así como para crear el propio sitio web de VisualWADE [27]. En esta sección se presentan algunas de las características de los sistemas que se han diseñado con VisualWADE dentro de estas empresas.

En SUMA gestión tributaria, se han desarrollado varias aplicaciones web, la mayoría de ellas son orientadas a datos. Entre ellas, destacamos por orden cronológico, una gestión de inventario, la intranet de SUMA y el portal de Internet. La gestión de inventario es una aplicación para administrar el material inventariable informático que está repartido a lo largo de sus 250 oficinas de la provincia de Alicante. Realmente, gestión de inventario puede verse como una aplicación de escritorio tradicional con una interfaz de usuario web. Ésta fué la primera aplicación que se diseñó completamente con VisualWADE y también sobre la que más errores de diseño se cometieron. El motivo fué que se estuvo muy influenciado por el tipo de interfaces de usuario que estaban acostumbrados a utilizar en la empresa y se nos forzó a realizar un diseño "antinatural" para un entorno web. Se utilizaron pocos patrones de visualización web con lo cual la navegación no era intuitiva y estaba dirigida por el conjunto de servicios que se ofrecían desde la interfaz al igual que en sus aplicaciones tradicionales. Gestión de inventario tenía más de 150 servicios disponibles en la aplicación. La experiencia de este desarrollo hizo madurar de forma paralela tanto al equipo SUMA como al nuestro y evidenciar que diseñando de esa forma no se conseguía aprovechar al 100% las bondades de los entornos web.

El segundo proyecto realizado en SUMA fué su intranet. Básicamente, la intranet proporciona la funcionalidad necesaria para manejar el conjunto de novedades, eventos, herramientas y documentos internos de la empresa a través de una interfaz web intuitiva y de fácil acceso para los usuarios. Es la primera aplicación que se diseñó para varios perfiles de usuario, entre ellos el de administrador cuya labor fundamental es fiscalizar la información que desde los distintos departamentos se va introduciendo en el sistema. La intranet de SUMA da servicio en estos momentos a más de 400 usuarios y al tercer mes de funcionamiento se había incrementado las visitas a la intranet en más de un 300% con respecto al mismo periodo del año anterior.

El último proyecto realizado es el portal de SUMA. Con un diseño de navegación parecido al de la intranet, esta aplicación ofrece nuevos e interesantes servicios a los ciudadanos y ayuntamientos. La Figura 8 muestra el aspecto del portal. En su diseño se ha utilizado algunas de las características avanzadas que proporciona VisualWADE como el soporte multiidioma y la generación de la interfaz basada en plantillas. De

hecho, el diseño gráfico fue desarrollado por una empresa externa subcontratada. Posteriormente, se adecuó este diseño gráfico al modelo de navegación generado.

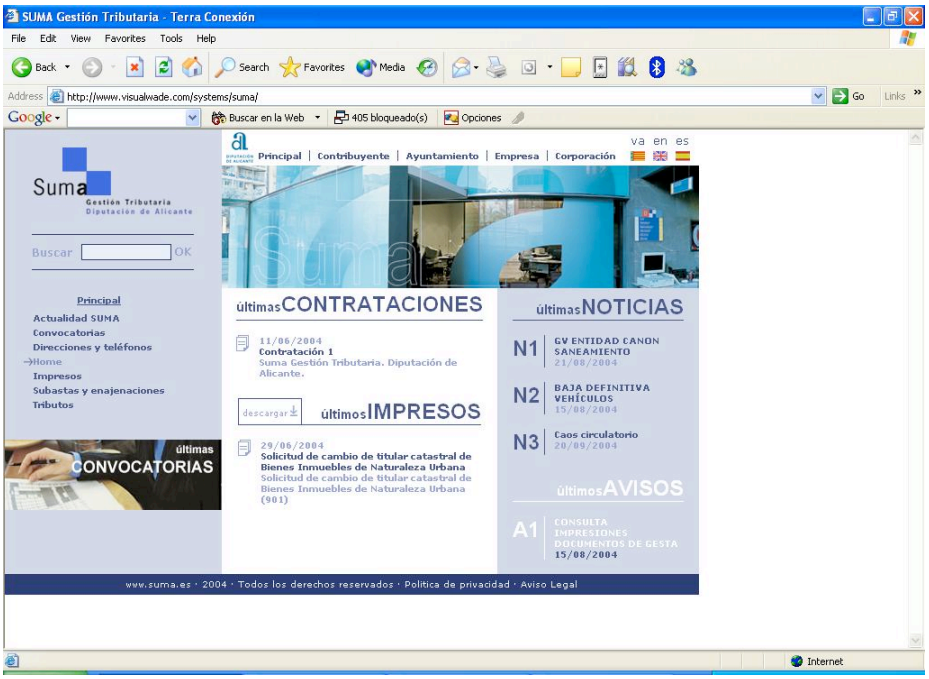


Figura 8: SUMA gestión tributaria

Respecto a la Caja de Ahorros del Mediterráneo, el uso de VisualWADE fué un tanto más limitado. Fundamentalmente, por los motivos de seguridad que se describen en la sección 5.2. Coordinados con el equipo de CAM directo, cuya responsabilidad es mantener el conjunto de aplicaciones para dar servicio bancario a través de Internet, se utilizó VisualWADE para diseñar exclusivamente la navegación de la interfaz de usuario del propio sistema CAM directo. La Figura 9 muestra la página de entrada a este servicio. En este caso, no se hizo uso alguno de las capacidades de VisualWADE para generar servicios predefinidos, puesto que la funcionalidad ya existía y había que simplemente invocarla desde en entorno de navegación. Para ello, se hizo uso de la generación de servicios web que "encapsulaban" las llamadas a las distintas operaciones necesarias a través de una especificación WSDL. Lo que más trabajo nos llevó fue adecuar las devoluciones de parámetros de los servicios al motor de navegación para su posterior presentación en la interfaz web. Lamentablemente, esta tarea de integración con servicios web todavía tiene que hacerse artesanalmente.

La experiencia fue muy positiva, pero también se evidenció que se pierde mucho potencial sobre el uso de VisualWADE, ya que sólo se utilizaron algunas de las características del entorno como el generador de interfaz. Las necesidades de privacidad y seguridad que exigen los entornos bancarios obligan a que se modifiquen los entornos de diseño web para que traten como elemento de primer nivel estos

requisitos de seguridad. En este sentido, se ha creado un grupo de trabajo con el equipo de CAM directo para identificar y proponer el conjunto de primitivas de modelado que se deben incorporar en un modelo de seguridad de aplicaciones web.

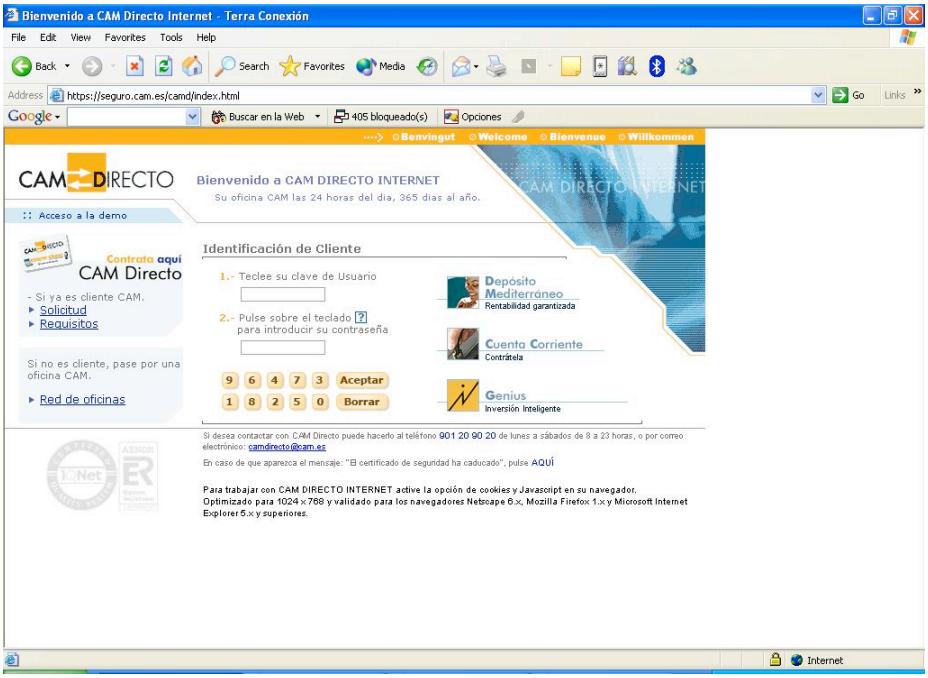


Figura 9: Sistema CAM directo de la Caja de Ahorros del Mediterráneo

La tramitación telemática de proyectos de Ingeniería Industrial para la obtención de visados electrónicos ha sido otro sistema en el que se ha aplicado VisualWADE con éxito. El sistema PROVE (PROyecto Visados Electrónicos), hace un uso intensivo de servicios web para ofrecer desde el entorno de navegación funcionalidades desarrolladas por terceros. Un ejemplo de estas funcionalidades son los servicios de firma digital proporcionados por la infraestructura de clave pública de la Generalitat Valenciana [28].

PROVE (ver Figura 10), es el primer sistema desarrollado con VisualWADE que conecta tres sistemas de información distintos; el servidor web del visados, la CAM y la Consellería de Industria. Conceptualmente, un Ingeniero Industrial puede, desde un único punto de entrada, realizar la solicitud de visado de un proyecto, pagar las tasas de gestión de este proyecto a través del sistema CAM directo y remitirlo automáticamente a la Consellería para su registro. Al igual que en el caso de SUMA, la integración con servicios web se ha tenido que realizar artesanalmente. Este es, por tanto, un tema pendiente de mejorar en futuras versiones de la herramienta.

PROVE da servicio actualmente a más de 1.350 colegiados tramitando aproximadamente más de 15.000 proyectos cada año.

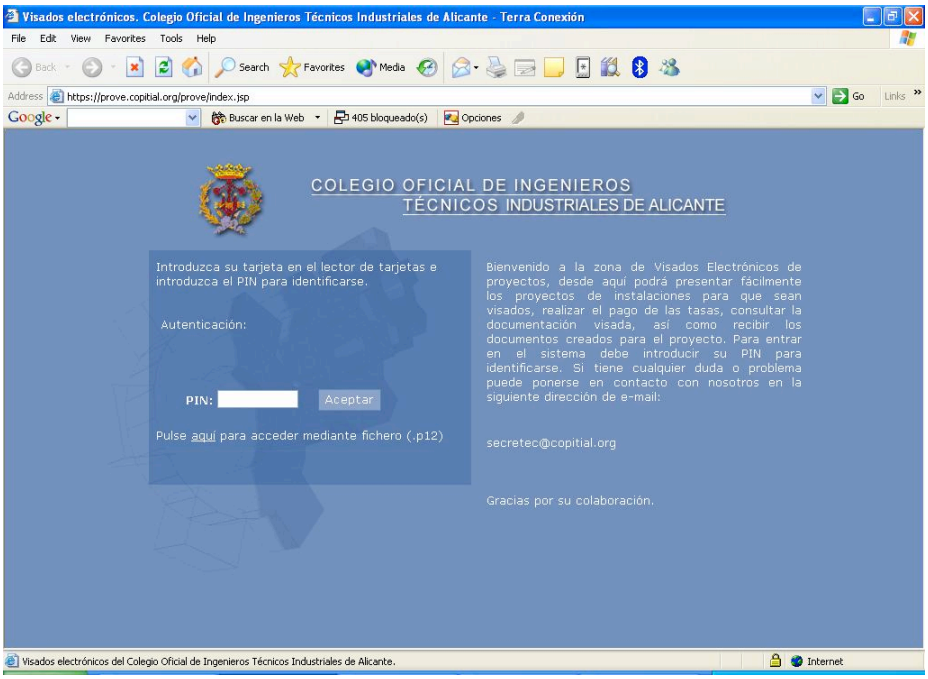


Figura 10: Visados Electrónicos en el Colegio de Ingenieros Técnicos Industriales

Por último, se ha realizado el diseño conceptual y generación del sitio web de la herramienta VisualWADE que puede verse en la Figura 11. El portal VisualWADE también está basado en plantillas y proporciona un conjunto de funcionalidades de navegación muy interesantes. Todas estas funcionalidades han sido generadas completamente haciendo uso de las propiedades de VisualWADE. Por ejemplo, la posibilidad de registrarse en el sitio web para acceder a la descarga gratuita de la herramienta, o de participar en foros de discusión o incluso de ejecutar una aplicación ejemplo de gestión de tareas de proyectos.

A diferencia de los sistemas que se han presentado anteriormente, en el portal VisualWADE no se hace uso de ninguna funcionalidad de terceros. Se ha seguido la filosofía de modelar y generar todo aquello que sea útil para el portal. En este sentido, podríamos haber utilizado cualquier software de gestión de foros gratuito que hay disponible en Internet, pero se prefirió hacerlo desde cero para demostrar la potencia y facilidad de uso del entorno.

El portal de VisualWADE que lleva abierto desde noviembre 2004, ha manejado una tasa de transferencia de más de 40 Gb en este mes y se han producido ya más de 1000 descargas de la herramienta desde lugares de todo el mundo.

Figura 11: Sitio Web de la herramienta VisualWADE

Todos estos sistemas que se han presentado brevemente constituyen un hecho real sobre la utilidad de los entornos de desarrollo web avanzados. Nuestro propósito actual es hacer llegar VisualWADE a la mayor cantidad de usuarios interesados posibles con el objetivo de explorar nuevos usos y experiencias de desarrollo por parte de terceros. Estamos seguros que esta nueva fase sacará a la luz interesantes experiencias todavía por descubrir.

A continuación se presentan algunas de las experiencias que hemos vivido en el desarrollo de estos sistemas de información web descritos.

5.2 Experiencias en el desarrollo de aplicaciones reales

En esta sección se proporcionan algunas de las experiencias del equipo VisualWADE en la aplicación de la herramienta para la resolución de casos reales. Como denominador común, todos estos casos de estudio necesitaban soluciones basadas en web para sus sistemas de información. El 85% de estas soluciones no requerían desarrollos nuevos, sino adaptaciones de sistemas de información existentes al nuevo entorno de la web. Mientras que tan sólo el 15% podían considerarse como nuevos desarrollos, aunque en realidad eran pequeñas extensiones sobre funcionalidades de las que no se disponía en ese momento.

Como se ha comentado anteriormente, VisualWADE promociona por un lado operaciones predefinidas para dar soporte al diseño de aplicaciones web de uso intensivo de datos (*data-intensive*). Este tipo de operaciones han demostrado ser suficientes para proporcionar lógica de negocio básica en aplicaciones como la gestión de inventario o la intranet de SUMA. De hecho, gran parte de las aplicaciones web que demandan las empresas son orientadas a datos. Por tanto, resulta particularmente importante proporcionar servicios CRUD para la gran mayoría de entornos web que demandan las empresas, sobre todo aquellas que poseen sistemas de información corporativos.

Por otro lado, la posibilidad de especificar y usar servicios web en VisualWADE ha facilitado el mecanismo de integración de funcionalidades que, en forma de software heredado (*legacy software*), se necesitaban reusar en el nuevo entorno. Por motivos relacionados con la seguridad, privacidad y complejidad entre otros, estas funciones heredadas no podían reescribirse de nuevo sino que debían ser integradas en las nuevas soluciones, ya que gran parte de estas empresas han realizado inversiones muy fuertes en el pasado para que sus sistemas de información alcancen la madurez y fiabilidad deseada. Por tanto, lo que éstas necesitan no son herramientas que les faciliten la construcción de software web desde cero, sino herramientas que les faciliten la migración de sus sistemas a los nuevos entornos preservando la funcionalidad existente y altamente fiable.

Respecto a las primitivas y patrones de navegación, nuestra experiencia muestra la necesidad de un uso intensivo de estructuras tipo *index*, *show all* y *guided tours* en las aplicaciones diseñadas y generadas. Además, el gran volumen de información que se maneja en ellas requiere presentar a menudo la información en trozos. Por tanto, se ha mostrado imprescindible ofrecer constructores para paginar los resultados. En VisualWADE tales constructores forman parte implícita de las primitivas de navegación, pudiendo el diseñador fijar la propiedad de paginar o no y definir cuántos objetos por página.

También se han identificado algunas carencias sobre primitivas de navegación particularmente útiles que por desgracia VisualWADE no soporta en este momento. Entre ellas, los *index* anidados y la selección de atributos múltiple. Se ha detectado que con estas dos primitivas de navegación se podrían cubrir de forma satisfactoria algunas situaciones de modelado pendientes que no se han podido especificar o que se han tenido que especificar de otra forma menos eficiente.

El entorno de animación proporcionado dentro del diagrama de presentación se muestra altamente productivo en varios sentidos. Por un lado, ayuda a disminuir la curva de aprendizaje de la herramienta en un 25%. Especialmente, sobre los conceptos del diagrama de navegación, ya que desde el entorno de animación se puede ver de un simple vistazo cual es el efecto de una especificación de modelado sobre la interfaz de usuario final. Por otro lado, la velocidad de desarrollo se ve incrementada en diez veces al proporcionar compiladores de modelos específicos que reproducen fielmente la apariencia y el comportamiento de la interfaz animada.

En todas las empresas en donde se ha aplicado VisualWADE se han tenido que realizar periodos de formación del personal informático. Nuestra experiencia en este sentido, demuestra que se requiere un reciclaje importante sobre gran parte del personal, especialmente respecto a técnicas de análisis y diseño orientado a objetos, lo cual nos resultó bastante sorprendente. El personal informático de estas empresas posee un conocimiento adecuado de modelado de datos, y por tanto entender o especificar un diagrama de clases no suponía mucho problema. Sin embargo, tuvimos que hacer esfuerzos más intensos en explicar el diagrama de navegación y sobre todo sus conceptos más avanzados como el uso de OCL para especificar filtros y precondiciones sobre enlaces y servicios. En el caso de la CAM, esta situación no se dió y la transferencia de conocimiento fue mucho más ágil debido a la alta formación de su personal.

Otro aspecto que se ha aprendido es que resulta muy importante que los modelos de navegación tengan en cuenta aspectos de tiempo sobre la información por la que se navega. Por desgracia, la gran mayoría de los métodos y herramientas de diseño web existentes excepto Ariadne (véase capítulo 2), no tienen en cuenta esta dimensión a la hora de especificar un espacio de navegación. En el proceso de modelado del proyecto SUMA intranet, se necesitaban soportar requisitos de navegación para mantener el periodo de caducidad de las noticias y eventos que se generaban en la propia intranet y nos dimos cuenta de que se necesitaba alguna formar tratar el tiempo en el modelo de navegación de VisualWADE. En este caso, se optó por una solución basada en extender el lenguaje OCL de VisualWADE con primitivas de tiempo. Esto, permitió por ejemplo poder contemplar en el entorno operaciones entre fechas, etc…

La experiencia de trabajar con VisualWADE en el equipo de CAM directo ha sido muy productiva. En especial, para aprender restricciones del entorno CAM y situaciones especiales que hubieran sido imposible darse cuenta sin esta experiencia. Por ejemplo, a la CAM no le gustó que los compiladores de modelos de VisualWADE produjeran un esqueleto fijo de generación. Ellos argumentaban que cualquiera que conociera como se comunican los distintos artefactos generados por VisualWADE entre sí, disponía de una información muy valiosa para atentar contra la seguridad del código generado. Por este motivo, nuestros esfuerzos para mejorar los compiladores de modelos van dirigidos a proporcionar mecanismos basados en MDD/MDA para producir distintos esqueletos de generación en base a un conjunto de plantillas de arquitectura.

6 Conclusiones

Los desarrolladores web deben mejorar la productividad y calidad para satisfacer las necesidades del mercado y reducir los tiempos de entrega y costes. Lamentablemente, los métodos y técnicas estándar que se utilizan para el desarrollo web todavía presentan varias deficiencias:

- Los modelos y herramientas de análisis y diseño carecen de conceptos adecuados para capturar las propiedades de desarrollo en este tipo de entornos como consecuencia la mayoría del código de la aplicación es escrito a mano y difícil de reutilizar.
- La documentación es escasa y de baja calidad, especialmente para la interfaz de usuario.
- Los costes y el tiempo de desarrollo son difíciles de predecir y rápidamente escapan de nuestro control durantes las fases de mantenimiento y evolución de la aplicación.

Todavía las herramientas de desarrollo web limitan su campo a la fase de implementación o están centradas en soluciones verticales concretas (a menudo basadas en componentes) que son difíciles de personalizar y encajar juntas.

Desde nuestra experiencia, el uso de técnicas CAWE (*Computer-Aided Web Engineering*) en el diseño/implementación de aplicaciones web proporciona los siguientes beneficios:

- *Incremento de la productividad*: hasta casi un 80% gracias a los potentes compiladores de modelos que generan el código de forma automática a partir de la especificación conceptual.
- *Prototipado rápido*: de uno a tres días son suficientes para entregar un prototipo de alta calidad. De esta forma el "*time to market*" se ve mejorado considerablemente.
- *Satisfacción del cliente*: el intercambio de información con el cliente es mucho más ágil y la respuesta ante cambios exigidos puede mostrarse en poco tiempo gracias a prototipado rápido. De esta forma se consigue transmitir confianza al cliente captando su fidelidad.
- *Disminución de la curva de aprendizaje*: en cinco días un diseñador puede producir aplicaciones web completas de alta calidad sin necesidad de conocimientos de programación.
- *Esfuerzo repartido a lo largo del ciclo de vida*: la posibilidad de dedicar más tiempo al usuario se traduce en un incremento de la calidad de la aplicación respecto a los requisitos del sistema.

Como consecuencia de estas reflexiones, y de nuestra experiencia a lo largo de los últimos cuatro años, creemos que en un futuro no muy lejano empezaran a aparecer las primeras herramientas de desarrollo que ofrezcan al mismo tiempo:

- La posibilidad de modelar visualmente una aplicación web haciendo uso de constructores de modelado de alto nivel, junto con compiladores de modelos que faciliten la generación automática de código portable y eficiente a partir de la especificación conceptual capturada.
- Soporte de todo el ciclo de vida de la aplicación, incluyendo, análisis, prototipado, diseño, despliegue y mantenimiento.

- Integración de módulos de lógica preexistentes (*legacy software*) en un proceso de desarrollo optimizado.

VisualWADE, en su estado actual proporciona alguna de estas demandas y actualmente se está utilizando en la resolución de casos reales complejos en organismos como SUMA Gestión Tributaria (Diputación de Alicante), Caja de Ahorros del Mediterráneo y el Colegio de Ingenieros Técnicos Industriales de la Provincia de Alicante. VisualWADE y OO-H están en continua evolución, nuestro próximo reto es integrar en el entorno el nuevo paradigma de desarrollo basado en MDD/MDA. Desde estas líneas invitamos al lector interesado a que se descargue una versión de la herramienta para comprobar las bondades de los entornos avanzados de desarrollo web.

Agradecimientos

Este capítulo ha sido subvencionado por el Ministerio de Educación y Ciencia a través del proyecto METASIGN de referencia TIN2004-00779.

Referencias

1. P. Atzeni, G. Mecca, and P. Merialdo. Design and Maintenance of Data-Intensive Web Sites. In Advances in Database Technology - EDBT'98, pages 436–449, 03 1998.
2. C. Cachero. The OO-HMethod Pattern Catalog. Technical report, Universidad de Alicante, 12 1999.
3. C. Cachero, J. Gómez. Advanced Conceptual Modeling of Web Applications: Embedding Operation Interfaces in Navigation Design. Actas de las VII Jornadas de Ingeniería del Software y Bases de Datos (JISBD). Pags. 235-248. El Escorial (Madrid) 2002.
4. S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. In Position Paper, Web Engineering Workshop, WWW9, 05 2000.
5. eXtensible Markup Language (XML). http://www.w3.org/XML/.
6. F. M. Fernández, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In Proceedings of ACM SIGMOD International conference on Management of data, pages 414–425, 10 1998.
7. P. Fraternali and P. Paolini. A Conceptual Model and a Tool Environment for Developing more Scalable, Dynamic, and Customizable Web Applications. In Advances in Database Technology - EDBT'98, pages 421–435, 1998.
8. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.
9. F. Garzotto and P. Paolini. HDM A Model-Based Approach to Hypertext Application Design. ACM Transactions on Information Systems (TOIS), 11(1):1–26, 01 1993.
10. D.M. Germán and D.D. Cowan. Three Hypermedia Design Patterns. In Proceedings of the HT99 Workshop on Hypermedia Development: Design Patterns in Hypermedia, 02 1999.

11. J. Gómez, C. Cachero. OO-H: Extending UML to Model Web Interfaces. Information Modelling for Internet Applications. Pags. 144-173. Idea Group Publishing, 2002.
12. T. Isakowitz, E. A. Stohr, and V. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. CACM: Communications of the ACM., pages 34–44, 08 1995.
13. S. McGrath. XML by Example. Building ecommerce Applications. Prentice Hall, 1998.
14. G. Mecca, P. Merialdo, P. Atzeni, and V. Crescenzi. The ARANEUS Guide toWeb-Site Development. Technical report, Universidad de Roma, 03 1999.
15. M.Nanard, J. Nanard, and P. Kahn. Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates. In HYPERTEXT '98. Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems, pages 11–20, 1998.
16. OMG Unified Modeling Language Specification Version 1.3. http://www.omg.org/cgi-bin/doc?ad/99-06-08.pdf, 06 1999.
17. P. Paolini and F. Garzotto. Design Patterns for WWW Hypermedia: Problems and Proposals. In Proceedings of the HT99 Workshop on Hypermedia Development: Design Patterns in Hypermedia, 02 1999.
18. G. Rossi, D. Schwabe, and A. Garrido. Design Reuse in Hypermedia Applications Development. In Proceedings of the eight ACM conference on HYPERTEXT '97, pages 57–66, 1997.
19. D. Schwabe, G. Rossi, and D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In Proceedings of the the seventh ACM conference on HYPERTEXT '96, page 166, 1996.
20. Jos Warmer and Anneke Kleppe. The Object Constraint Language. Precise Modeling with UML. Addison-Wesley, 1998.
21. WebRatio Web Site http://www.webratio.com.
22. S. Montero, P. Díaz, I. Aedo. AriadneTool: A Design Toolkit for Hypermedia Applications. J. Dig. Inf. 5(2): 2004.
23. A. Knapp, N. Koch, F. Moser, G. Zhang. ArgoUWE: A CASE Tool for Web Applications. EMSISE03, 14 pages, online publication at http://www.pst.informatik.uni-muenchen.de/~kochn
24. SUMA Gestion Tributaria. http://www.suma.es
25. Caja de Ahorros del Mediterráneo. http://www.cam.es
26. PROVE Proyecto Visados Electronicos. http://www.copitial.org
27. VisualWADE. http://www.visualwade.com
28. PKI GVA. http://www.pkigva.es

4 MIDAS: Una Aproximación Dirigida por Modelos para el Desarrollo Ágil de Sistemas de Información Web

Belén Vela, Paloma Cáceres, Valeria de Castro y Esperanza Marcos

Grupo Kybele
Universidad Rey Juan Carlos
Madrid, España
{belen.vela,paloma.caceres,valeria.decastro,esperanza.marcos}
@urjc.es

4.1 Introducción

En la última década, la web se ha consolidado como uno de los principales medios para compartir y difundir información a nivel mundial. La gran mayoría de las empresas y organizaciones han tenido que adaptarse a este nuevo entorno, inicialmente publicando su información en la web, y en la actualidad extendiendo su capacidad de negocio mediante la utilización de este medio. Ha surgido así la necesidad de técnicas y metodologías de Ingeniería de Software adaptadas al desarrollo específico de Sistemas de Información Web (SIW).

Las primeras propuestas metodológicas eran, en general, adaptaciones de metodologías clásicas enfocadas a un tipo concreto de desarrollo. Así los trabajos propuestos en [20,26,30,47,48] surgen del campo de la hipermedia y multimedia, mientras que otras, como [8,23,24,39] proceden principalmente de aproximaciones para el desarrollo de Bases de Datos (BD). También surgieron extensiones de metodologías de desarrollo de propósito general como [18,27], así como metodologías genéricas que permitían abordar diversos tipos de aplicaciones mediante la adaptación de su proceso de desarrollo [11,15]. En la actualidad, se habla de una nueva área de trabajo que se denomina Ingeniería de la Web (*Web Engineering*) y cuyas aportaciones surgen, directamente, con el objetivo de facilitar el desarrollo sistemático y (semi-)automático de SIW adaptando al desarrollo web prácticas de la Ingeniería del Software.

En el marco anteriormente descrito surge MIDAS como una aproximación metodológica al desarrollo de BD web [36]. Aunque inicialmente MIDAS comenzó como una adaptación de metodologías clásicas de diseño de BD [35,37] al desarrollo de aplicaciones web, ya desde prácticamente sus orígenes se planteó la necesidad de especificar una metodología genérica y fácilmente adaptable a cualquier tipo de SIW. Para ello, MIDAS define: a) un proceso de desarrollo ágil [13]; y b) una arquitectura basada en modelos [14] en la que se identifican tres aspectos a considerar en el desarrollo de un SIW: el contenido, el hipertexto y el comportamiento.

a) Se propone un `proceso de desarrollo ágil` debido a la necesidad de celeridad en los desarrollos de SIW. Como las metodologías tradicionales generalmente son excesivamente burocráticas, se ha pasado, en la mayoría de los casos, a no utilizar ningún método de trabajo específico y a trabajar `a destajo` con el único y erróneo objetivo de ahorrar tiempo y dinero. Aplicar cierto grado de disciplina ayudará en el proceso de desarrollo y, por lo tanto, la utilización de un

proceso ágil puede ser más adecuada que un proceso demasiado rígido y estricto; en cualquier caso, siempre será mejor utilizar un proceso ágil que no utilizar ninguno. Aunque parece claro que los procesos ágiles, también llamados adaptativos, no son adecuados para el desarrollo de cualquier tipo de aplicaciones [25], si se puede afirmar que son especialmente útiles para desarrollos web.

Los criterios principales para la elección de un proceso de desarrollo ágil frente a un proceso de desarrollo tradicional son:

- Los requisitos software son desconocidos o variables.
- Se necesita una generación rápida de prototipos.
- Se necesitan entregas de versiones del producto software previas a la entrega final, para que el cliente se sienta más cercano al producto desde los comienzos del proyecto.
- Existe una alta probabilidad de realizar cambios durante el ciclo de desarrollo del software.
- Los ciclos de desarrollo no demasiado largos.

Por otra parte, las características más representativas de los desarrollos web, como ya se mencionó en el capítulo 1, son:

- Se necesita una disponibilidad rápida del producto software en la web.
- Los ciclos de desarrollo son generalmente más cortos que los desarrollos tradicionales.
- Los requisitos de usuario son desconocidos, excepto para los sistemas de información y las aplicaciones de Intranet.
- Existe la creencia equivocada de que estos desarrollos son muy sencillos.

Es fácil observar que existe una significativa relación entre las características de los desarrollos de SIW y los factores que determinan el uso de un proceso de desarrollo ágil, que se resume de la siguiente manera:

- Para satisfacer la rápida disponibilidad del producto software en la web es conveniente un proceso de desarrollo que garantice entregas de versiones del producto software previas a la entrega del producto final. La generación de prototipos supone un acercamiento entre cliente y producto, agilizando así la validación de los productos intermedios y, por lo tanto, agilizando también dichas entregas previas a la entrega final.
- Dado que parece generalmente aceptado que los ciclos de desarrollo web son más cortos que los tradicionales y puesto que los procesos software ágiles son adecuados para los ciclos de desarrollo cortos, la relación de conveniencia en este punto también es clara.
- Los requisitos de usuario de los productos web son desconocidos, excepto para los casos de Intranet. Existen ciertos criterios o factores generalmente aceptados acerca de facilidades y consideraciones a tener en cuenta en este tipo de desarrollos, pero desde luego no cabe duda de que el usuario de un SIW puede ser cualquier persona del mundo con acceso a Internet. Por este motivo, los procesos de desarrollo ágiles también son adecuados, puesto que aportan ventajas significativas sobre los tradicionales cuando tienen que tratar con requisitos desconocidos o variables, lo que además implicará la necesidad de realizar cambios a lo largo de todo el ciclo de desarrollo.

- Generalmente los clientes de productos software web, desconocen la profundidad y complejidad de este tipo de desarrollos, por lo que se va a tender a subestimar el esfuerzo técnico de su desarrollo. Dicha subestimación repercute en una toma de decisiones con cierta ligereza por parte de los clientes. Hecho que conlleva la necesidad de tener que realizar cambios durante todo el desarrollo. En este sentido siempre será de mayor ayuda un proceso adaptativo frente a un proceso predictivo.

Por todo ello, se puede concluir que la utilización de procesos ágiles es beneficiosa para el desarrollo de aplicaciones web.

b) El proceso de desarrollo, tal como se ha indicado anteriormente, se integrará en una `arquitectura dirigida por modelos`. A pesar de que las propuestas más `extremas` de procesos ágiles (v.g. *eXtreme Programming* (XP) [9]), proponen que la documentación sea directamente el código y eliminan de algún modo la necesidad de utilización de modelos, en este trabajo se propone una aproximación ágil más moderada. Si bien es verdad que el desarrollo de un SIW simple posiblemente podría llevarse a cabo sin necesidad de modelos y pasando directamente a codificar, no está tan claro que este tipo de construcción pudiera ser eficaz para el desarrollo de sistemas más complejos. El desarrollo dirigido por modelos facilita el análisis de aplicaciones complejas, favorece la tarea de documentación, mejorando de este modo el proceso de mantenimiento y evolución del software, y proporciona facilidades para la generación (semi-)automática de aplicaciones.

Además, no es incompatible en absoluto la integración de una arquitectura dirigida por modelos en un proceso de desarrollo ágil, si se es capaz de ver que los modelos no son otra cosa que lenguajes de programación de alto nivel, y que las herramientas de desarrollo MDA (*Model Driven Architecture*) son compiladores también de alto nivel. En este sentido, es posible aplicar técnicas de XP, como la programación por pares, directamente al desarrollo basado en modelos.

Por todo ello, en MIDAS se propone una arquitectura dirigida por modelos basada en MDA propuesta por el *Object Management Group* (OMG) [42] y un proceso de desarrollo basado en el uso de técnicas procedentes de metodologías ágiles como XP [9] y AM (*Agile Modeling*) [6]. Existen otros autores que también proponen la combinación de métodos ágiles y dirigidos por modelos para el desarrollo de SIW [5,49,53] y, de hecho, ya existe un término acuñado en el campo de la Ingeniería del Software que identifica este tipo de desarrollos como AMDD (*Agil Model Driven Development*).

En MIDAS también se propone el uso de estándares en el proceso de desarrollo de un SIW (UML [41], XML [55], SQL:2003 [22], WSDL [54], etc.), así como el uso de UML para el modelado del SIW independientemente del nivel de abstracción o del aspecto del sistema a modelar. Dado que UML no permite directamente representar todos los modelos necesarios para modelar un SIW, MIDAS incorpora algunas extensiones de UML existentes [28,31] y define, o adapta, otras nuevas, siempre que es necesario [34,35,52].

MIDAS propone modelar el SIW de acuerdo con tres aspectos básicos: contenido, hipertexto y comportamiento. Todos estos aspectos se consideran a nivel de **`Modelos Independientes de Plataforma`** (PIMs, *Platform Independent Models*) y **`Modelos Específicos de Plataforma`** (PSMs, *Platform Specific Models*). Además, se proponen modelos específicos para el nivel de **`Modelos`**

Independientes de Computación (CIMs, *Computation Independent Models*).

Este trabajo se centra en los aspectos de hipertexto y de contenido. Existen otros trabajos relacionados donde se describen con más detalle los aspectos relacionados con el comportamiento del sistema [33, 34]. Existe además un primer prototipo de una herramienta de desarrollo basado en MIDAS, cuya exposición queda fuera del alcance de este trabajo; un resumen de la misma puede verse en [50].

Para validar MIDAS, se han utilizado diferentes casos de estudio especialmente seleccionados para cada uno de los métodos y modelos a validar. En la actualidad, la metodología completa está siendo aplicada en un caso real: el desarrollo de un SIW para la gestión de imágenes médicas [2,3].

El resto del capítulo se estructura del siguiente modo: la sección 2 describe el marco metodológico, incluyendo el proceso de desarrollo y la arquitectura dirigida por modelos; las secciones 3 y 4 detallan los métodos específicos para el desarrollo del hipertexto y del contenido, respectivamente, incluyendo los modelos y las transformaciones entre los mismos; la sección 5 resume las principales lecciones aprendidas en la utilización de MIDAS; y en la sección 6 se finaliza con las principales conclusiones, así como los trabajos que se están llevando a cabo en la actualidad.

4.2 Desarrollo ágil basado en modelos en MIDAS

Como se ha comentado previamente, MIDAS propone la integración de:

- Una arquitectura dirigida por modelos basada en: a) MDA (*Model Driven Architecture*); y b) los diferentes aspectos de un SIW.
- Un proceso de desarrollo que propone el uso de técnicas procedentes de metodologías ágiles como XP [9] y AM [6].

4.2.1. Arquitectura dirigida por modelos

La arquitectura dirigida por modelos de MIDAS considera dos dimensiones ortogonales: a) la primera dimensión está basada en la propuesta de MDA de modelar el sistema en base a CIMs, PIMs y PSMs; y b) la segunda considera los diferentes aspectos que han de ser tenidos en cuenta en el desarrollo de un SIW; para ello, y como punto de partida, se han utilizado como referencia los modelo arquitectónicos de n-capas, considerando así los aspectos de contenido, hipertexto y comportamiento del sistema.

a. MDA – Model Driven Architecture.

MDA [42,43] es una arquitectura dirigida por modelos para el desarrollo de software, definida por el OMG. Su finalidad es recoger las especificaciones de un sistema generando, como productos, un conjunto de modelos formales, que puedan incluso llegar a ser comprendidos por un ordenador. Se propone así la realización de modelos a diferente nivel de abstracción, y de dependencia respecto de la plataforma específica de implementación, además de un conjunto de reglas de transformación *(mappings)* entre dichos modelos.

Los modelos propuestos son: **Modelos Independientes de Computación** que son modelos del más alto nivel de abstracción y que definen

el contexto del sistema; Modelos Independientes de Plataforma que proporcionan la especificación formal de la estructura y función del sistema, sin tener en cuenta aspectos técnicos específicos de cualquier tecnología de implementación; y Modelos Específicos de Plataforma que proporcionan modelos en términos de constructores de implementación que están disponibles en una tecnología específica (v.g. .NET, servicios web, etc.).

Las reglas de transformación se aplican para transformar: PIM en PIM, que van generalmente asociadas a los pasos que se suceden entre el modelado de la especificación, análisis y diseño; PIM en PSM, que se realizan cuando el PIM está suficientemente refinado y se transforma en un modelo dependiente de la infraestructura final de ejecución, de modo que un PIM se transforma en uno o varios PSM; PSM en PSM, que son necesarios para la realización y despliegue de componentes; PSM en PIM, que permiten la abstracción de modelos a partir de implementaciones específicas de una plataforma y dependientes de una tecnología concreta.

Al igual que en MDA, en MIDAS se proponen diferentes CIMs, PIMs y PSMs, así como reglas de transformación entre modelos, tanto en el mismo nivel como entre diferentes niveles. Además, MIDAS propone modelar el sistema en base a los aspectos principales de un SIW.

b. Principales aspectos en el modelado de un SIW

Partiendo de las propuestas de MDA, ya se tiene una de las dimensiones de la arquitectura dirigida por modelos de MIDAS. Sin embargo, es necesario además considerar los diferentes aspectos de un sistema, con el fin de poder modelarlos independientemente [32]. Para determinar los aspectos básicos a considerar en el desarrollo de un SIW, se ha estudiado: a) los aspectos de estructura y comportamiento, y los niveles de presentación, hipertexto y contenido definidos en [46] como básicos para el desarrollo de un SIW (ver Figura 4.1); y b) los modelos arquitectónicos de n-capas, como .NET o J2EE, ya que son los más comúnmente utilizados para los desarrollos web. En la actualidad, en MIDAS se contemplan los tres aspectos correspondientes a las tres capas más comúnmente aceptadas: persistencia, interfaz gráfica de usuario y lógica de negocio. Por uniformidad con la terminología usada en el campo de la Ingeniería de la Web [46], estos aspectos serán denominados contenido, hipertexto y comportamiento, respectivamente. Sin embargo, esta arquitectura cuenta con la ventaja de ser fácilmente escalable, permitiendo la incorporación, siempre que así se considere, de un nuevo aspecto o de un nuevo nivel.

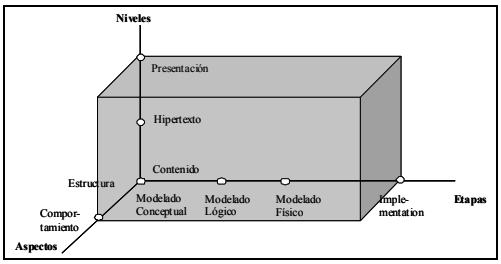


Fig. 4.1.- Requisitos de modelado de un SIW [46]

La Figura 4.2 muestra la arquitectura dirigida por modelos de MIDAS simplificada, donde se proponen los CIM, comunes a todo el sistema, así como los diferentes PIM y PSM para representar los aspectos de hipertexto, contenido, comportamiento.

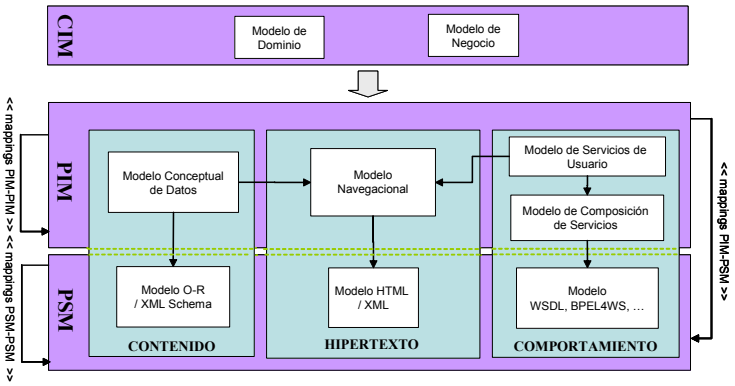


Fig. 4.2.- Arquitectura dirigida por modelos de MIDAS simplificada

Existiría además una tercera dimensión que incluiría otros aspectos a tener en cuenta en el desarrollo de un SIW, como son la arquitectura del sistema o la seguridad, que son ortogonales a los presentados en la Figura 4.2. La Figura 4.3 muestra la arquitectura con las tres dimensiones mencionadas.

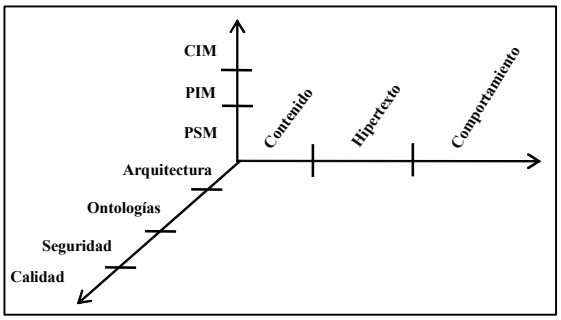


Fig. 4.3.- Dimensiones a considerar en un desarrollo de un SIW

4.2.2. Proceso de desarrollo

El proceso de desarrollo de las metodologías tradicionales impone una disciplina de trabajo con el fin de conseguir que el proceso de desarrollo software sea lo más predecible posible. Para ello, se hace especial hincapié en la planificación completa del trabajo completo a realizar y sólo una vez que está todo detallado, comienza el ciclo de desarrollo del producto software.

Estas metodologías han recibido diversas críticas. Quizás la más común de ellas hace referencia a su carácter excesivamente burocrático, lo que, tal y como se afirma en [21,25], en muchas ocasiones ha llevado a identificarlas con el nombre de `metodologías monumentales`. Por otra parte, las metodologías tradicionales no se adaptan adecuadamente a los cambios por lo que generalmente, no son especialmente adecuadas cuando se trabaja en entornos donde los requisitos, o bien no pueden predecirse, o bien pueden variar, como ocurre en el desarrollo de aplicaciones web. En los últimos años, y en contraposición con estas metodologías clásicas, ha aparecido un nuevo grupo de metodologías denominadas `ágiles`. Las metodologías ágiles buscan un equilibrio en la relación proceso/esfuerzo, de modo que proponen la aplicación de procesos de desarrollo sin hacer un excesivo esfuerzo en los aspectos más burocráticos de los mismos, como es el desarrollo de una exhaustiva documentación.

Las diferencias existentes entre ambos grupos de metodologías proceden de sus enfoques y objetivos. Como principales diferencias, Fowler [25] identifica las siguientes:

- Las metodologías ágiles son adaptativas más que predictivas. Las metodologías tradicionales potencian la planificación detallada de prácticamente todo el desarrollo software a largo plazo. Pero cuando se produce un cambio, toda esta planificación puede venirse abajo. Sin embargo, las metodologías ágiles proponen procesos que se adaptan y progresan con el cambio, llegando incluso hasta el punto de cambiar ellos mismos.
- Las metodologías ágiles están orientadas al personal más que orientadas al proceso. Intentan trabajar potenciando las características del personal asignado al desarrollo, de tal forma que permiten que la actividad de desarrollo software se convierta en una actividad grata e interesante.

En MIDAS, debido a las ventajas de los procesos ágiles para el desarrollo web, se ha definido un proceso de desarrollo basado en un modelo de proceso iterativo, incremental, adaptativo y con prototipado. Además incorpora técnicas de desarrollo orientadas al personal, procedentes de metodologías ágiles. Se ha propuesto que sea iterativo para garantizar la realimentación de información y de requisitos una vez iniciado el desarrollo, así como la validación continua del SIW. De este modo, cada iteración contempla ciclos de desarrollo completos y cortos, lo que permite obtener rápidamente una nueva versión con mejoras sobre las versiones anteriores. Se ha propuesto que sea incremental con la finalidad de obtener el sistema final tras la realización de diferentes etapas. Cada etapa está ligada a la realización de un determinado tipo de tareas. Al final de cada etapa se proporciona además, una versión estable del software, que gracias a esas tareas concretas, cuenta con unas características específicas. Esto permite entregas al cliente de forma rápida y ágil. Se ha propuesto que sea adaptativo para permitir su aplicación a diferentes tipos de desarrollos web. De esta forma, cada tipo de aplicación conllevará la ejecución de un determinado conjunto de tareas y pasará por unas determinadas etapas.

Por otra parte, y dado que en MIDAS también se apostaba por una arquitectura dirigida por modelos, ha sido necesario analizar la viabilidad de un proceso ágil y dirigido por modelos. En este sentido, según puede apreciarse en la Figura 4.4, los procesos ágiles y los procesos dirigidos por modelos no contemplan igualmente el espacio del problema y el espacio de la solución [53].

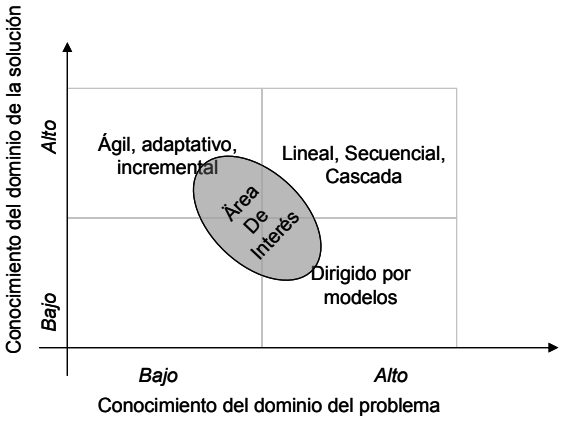


Fig. 4.4.- Relación entre el espacio del problema y de la solución

El área de interés de MIDAS, la parte sombreada de la Figura 4.4, se ha centrado en el acercamiento entre ambas propuestas, en base al estudio de cinco aspectos diferenciadores entre ambos procesos, indicados en la Tabla 4.1 y propuestos por [53].

Tabla 4.1. Aspectos diferenciadores entre procesos ágiles y procesos dirigidos por modelos

Aspectos	Ágiles	Dirigidos por modelos

Personas	Alta prioridad; se facilita relación cliente-desarrollador	No prioritario; el modelo del espacio del problema es la base de la discusión entre cliente-desarrollador
Proceso	Prioridad media; incremental y evolutivo	Tiende al proceso en cascada, poco incremental
Tecnología	Baja prioridad; sólo cobra importancia al final	Es relevante; se usa para la generación del software (usando un PSM)
Modelos	Artefacto secundario; se producen cuando es absolutamente necesario	Artefacto prioritario; fuente de la implementación
Software	Artefacto prioritario; es la única medida de progreso	Artefacto secundario; depende del espacio de la solución

En base a estos aspectos, y para realizar un acercamiento entre ambos tipos de procesos, en MIDAS se propone lo siguiente:

`Personas`. Al igual que en los procesos ágiles, se apuesta por darle una alta prioridad a la relación con el cliente. Aunque en los procesos dirigidos por modelos, el cliente entra a formar parte del proyecto a partir del modelo del espacio del problema, es muy importante que esa relación exista desde el principio: para establecer cuál es el espacio del problema, para definirlo y para utilizarlo posteriormente como base de la discusión entre el cliente-analista, entre el cliente-diseñador/arquitecto y entre el cliente-desarrollador. El cliente es parte fundamental durante todo el proceso, puesto que es la forma de garantizar que el producto final cumpla sus expectativas.

`Proceso`. El modelo de proceso propuesto en MIDAS tiene mucho que ver con el de los procesos ágiles puesto que es un modelo iterativo, incremental, adaptativo y con prototipado, lejano por tanto del modelo en cascada más comúnmente usado en los procesos tradicionales. El proceso iterativo e incremental aporta grandes ventajas puesto que permite la obtención de versiones del producto software antes de la entrega final del mismo. Ambler en [5] confirma además, la posibilidad de realizar un desarrollo iterativo e incremental, mediante un modelo de proceso ágil y dirigido por modelos.

`Tecnología`. En este aspecto la propuesta de MIDAS es un desarrollo basado en estándares, de modo que se permita la especificación de modelos independientes de plataforma. En cuanto a los modelos dependientes de plataforma, hasta este momento se han considerado: XML y tecnología Objeto-Relacional (OR) para el aspecto de contenido; XML y HTML para el aspecto de hipertexto; y arquitecturas orientadas a servicios para el aspecto de comportamiento.

`Modelos`. Los modelos son los artefactos que se generan y es la única documentación que se considera además del propio código. A diferencia de los procesos ágiles, esta propuesta considera que el conocimiento acerca del sistema y del negocio no puede mantenerse exclusivamente en la mente de los desarrolladores y del cliente. Según Atkinson y Kühne [7], es conveniente tanto para satisfacción del cliente como para el buen desarrollo del proyecto, que se deje constancia de ello por escrito con una notación concisa; en MIDAS se propone dejar constancia a través de modelos en notación UML y UML extendido.

`Software`. El objetivo de cada iteración de MIDAS es la obtención de un prototipo o versión del producto software a través de ciclos cortos de desarrollo, con

la finalidad de garantizar el progreso del software. Gracias a que el proceso es además incremental, el producto final se obtiene a través de versiones intermedias, que son operativas pero incompletas. Esto permite realizar entregas parciales del producto software al cliente de forma rápida, siempre previas a la versión final. Por lo tanto, sí que se dota de una alta prioridad también a este aspecto, tanto mediante la generación de prototipos, como de versión intermedias de software.

Por todo ello, en MIDAS se propone un proceso de desarrollo en el que se definen un conjunto de flujos de trabajo a llevar a cabo a lo largo del ciclo de desarrollo y, de forma ortogonal un conjunto de prácticas ágiles basadas principalmente en XP [9] y en AM [6]. Dicho proceso ágil de desarrollo aplicado sobre la arquitectura dirigida por modelos de MIDAS, da lugar a un desarrollo ágil y dirigido por modelos (AMDD) que facilita la generación de software de una manera iterativa e incremental, generando modelos y código casi simultáneamente. La Tabla 4.2 muestra un resumen de las actividades y flujos de trabajo asociados al proceso de desarrollo de MIDAS.

Tabla 4.2. Flujos de trabajo y actividades asociados al proceso de desarrollo de MIDAS

Flujos de trabajo	Actividades	Artefactos de Entrada	Artefactos de Salida
Requisitos	Modelar el contexto del SIW Capturar los requisitos Priorizar los requisitos Planificar los requisitos	Historia acerca del contexto del sistema Historias de usuario	Modelado de dominio y de negocio Planificación de la versión Tareas a realizar Arquitectura Requisitos agrupados en base a los aspectos Casos de prueba
Hipertexto	Modelar el hipertexto Desarrollar prototipos y/o una aplicación web estática	Planificación de la versión Requisitos del hipertexto Casos de prueba	Modelado del hipertexto Prototipos y/o aplicación web estática
Contenido	Modelar el contenido Desarrollar prototipos y/o de una aplicación web dinámica	Planificación de la versión Requisitos de contenido Casos de prueba	Modelado del contenido Prototipos y/o aplicación web dinámica
Comporta-miento	Modelar el comportamiento Desarrollo de prototipos y/o una aplicación web dinámica y orientada a servicios	Planificación de la versión Requisitos de comportamiento Casos de prueba	Modelado del comportamiento Prototipos y/o aplicación web dinámica y orientada a servicios

Las prácticas ágiles de MIDAS se basan en las prácticas de XP, pero ampliando su ámbito de aplicación. XP propone la `programación por pares` como una de sus prácticas más relevantes. Sin embargo, MIDAS propone en su lugar, el `desarrollo por pares`, es decir, que el personal trabaje por pares no sólo en las actividades de programación sino también durante las actividades de análisis, diseño, etc. De hecho, MIDAS se centra en el `ciclo de desarrollo`, es decir, en capturar requisitos, analizar, diseñar, implementar y probar el software, y no solamente en la etapa de implementación. Por lo tanto, MIDAS tampoco tiene en

cuenta sólo el código como documentación, sino también los modelos obtenidos tras la aplicación de las actividades previstas en el proceso. Por eso en MIDAS se propone centrarse en todo el `software` generado, y no sólo en el código.

Además, la `separación de aspectos`, presentada en la sección anterior en la arquitectura de MIDAS, en hipertexto, contenido y comportamiento, también tiene su repercusión en las prácticas propuestas. La separación de aspectos ofrece una serie de ventajas [32] en el desarrollo del software, puesto que generalmente permite incidir y detectar así las particularidades de cada uno de esos aspectos, de forma independiente. En MIDAS, la separación de aspectos se tiene en cuenta a lo largo de todo el ciclo de desarrollo.

Resumiendo lo mencionado en los dos párrafos anteriores, se puede decir que MIDAS dirige el desarrollo de SIW con tres principios en mente: un enfoque en el `ciclo de desarrollo` y no sólo en la etapa de programación; un enfoque centrado en el `software` y no sólo en el código; y la aplicación de la `separación de aspectos,` hipertexto, contenido y comportamiento, a lo largo de todo el ciclo de desarrollo.

A continuación, se presentan las prácticas propuestas en MIDAS, basadas en estos tres principios:

- Planificación basada en la separación de aspectos: permite una fácil estructuración y planificación del proyecto de desarrollo.
- Versiones pequeñas. MIDAS recomienda la generación de pequeñas versiones de software y de prototipos, facilitando así el acercamiento del producto final al cliente del desarrollo.
- Metáfora. MIDAS propone establecer la metáfora a partir de los modelos del dominio y del negocio a nivel de CIM. A partir de dichos modelos se establece un vocabulario con el que todo el personal del proyecto tiene que estar familiarizado, lo que fomenta el uso del mismo lenguaje que utiliza el cliente, favoreciendo su confianza y satisfacción.
- Tests. La definición y diseño de casos de prueba se ve muy beneficiada por la separación de aspectos, puesto que éstos necesitan clases de tests específicas. Así, para el aspecto de hipertexto es necesario probar, entre otras cosas, los enlaces entre páginas, y para el aspecto de contenido será necesario contemplar las altas y bajas de información.
- Diseño sencillo. En XP se apuesta por hacer `cosas sencillas`, es decir, pensar en realizar pequeños módulos de programación que tengan funciones lo más sencillas posibles. Dado que en MIDAS se proponen más actividades que la de programación, el objetivo entonces a perseguir, es facilitar la división del SIW en partes lo más sencillas posibles. Aquí se propone aplicar la separación de aspectos, puesto que será más fácil generalmente, diseñar el SIW teniendo en cuenta los diferentes aspectos a considerar [32].
- Refactorización. La refactorización en XP consiste en la reestructuración de código fuente, alterando su estructura interna pero sin modificar su comportamiento externo. Dado que para XP, el código es su documentación, esta reestructuración generalmente tiene lugar, cuando se considera que el código fuente podría mejorarse con el fin de poder leerlo mejor y así mejorar sus futuras actualizaciones. En este mismo sentido, MIDAS propone refactorizar todo el software, puesto que el software es la documentación de MIDAS. Entonces, en

MIDAS se propone refactorizar, no sólo el código, si no también los modelos y todo aquello que forme parte del software.

- Integración continúa. MIDAS propone la integración continua de todo el software y no sólo del código, lo que garantiza la consistencia del producto software que se vaya obteniendo a lo largo del proyecto.
- Propiedad colectiva del código. La propiedad colectiva del código en XP significa que la propiedad del código generado en un proyecto no es del individuo que generó ese código, si no de todo el proyecto. En XP se argumenta que si los desarrolladores tienen este sentimiento de propiedad colectiva, van a procurar que este código sea de una gran calidad. En MIDAS, esta práctica que se aplica sólo al código en XP, se aplica también a todo el software.
- Desarrollo por pares. En XP, se propone la `programación por pares` que significa que el personal del proyecto se divida en pares de programadores, con el fin de que las dos personas que forman un par, programen de forma conjunta y sobre un mismo equipo. Al haber dos personas trabajando de forma conjunta, se potencia el espíritu de superación porque cada una de esas personas, se intenta superar frente a la otra, aportando cada una lo mejor de sí. En MIDAS se propone que el personal trabaje por pares a través de todo el ciclo de desarrollo y no sólo durante la etapa de programación. El beneficio que se supone a la programación realizada de forma conjunta entre dos desarrolladores, como se propone en XP, en MIDAS se extrapola a cualquier actividad que se realice a lo largo de todo el ciclo de desarrollo. De hecho, en cuanto al desarrollo se refiere, los lenguajes han ido evolucionando desde el ensamblador hasta lenguajes de alto nivel, incluso, hoy en día, hasta lenguajes gráficos de modelado [29] como UML. Por tanto, si la programación por pares es posible, también es posible el modelado por pares.
- Uso de estándares durante el desarrollo. MIDAS propone el uso de estándares como UML, SQL:2003, XML, incluyendo XML Schema, XLink, WSDL, etc, a través de todo el ciclo de desarrollo.
- Cliente en el desarrollo o disponible vía web. En la actualidad, no es posible en muchos casos que el cliente pueda estar de forma continua en todo el proceso de desarrollo. Gracias a la tecnología web, la alternativa a tener al cliente en el lugar de desarrollo, es que esté disponible vía web.
- Software disponible en la web. Esta práctica propone que todo el software esté disponible a través de la web. De esta forma se garantizan dos cosas: por una parte la continua disponibilidad del software y, por otra, su consistencia.

4.3 Desarrollo del hipertexto en MIDAS: una aproximación basada en el usuario

Tal y como se ha dicho en la sección 4.2, en MIDAS se consideran tres aspectos fundamentales: contenido, hipertexto y comportamiento. Esta sección se centra en los modelos y el método propuesto para el modelado del hipertexto.

Las metodologías tradicionales para el modelado del hipertexto proponen, en general, el modelado del mismo partiendo del modelo de clases [8,11,28,30,31]. De este modo se obtienen modelos de hipertexto con una perspectiva estructural donde, en el mejor de los casos, los servicios y la funcionalidad asociada al sistema se añaden posteriormente al modelo `estructural` de hipertexto. A diferencia de estos

enfoques, MIDAS propone una nueva perspectiva `orientada a servicios de usuario` para el modelado del hipertexto [12,16]. Esta aproximación se basa en modelar el hipertexto en base a los servicios que el usuario requiere del sistema o, lo que es lo mismo, en las operaciones que el usuario quiere realizar en dicho sistema. Para ello, se parte de un modelo de caso de uso, al que se ha denominado `modelo de servicios de usuario,` que permite representar este tipo de servicios. Además, MIDAS propone modelar el proceso para la realización de cada servicio a través de un diagrama de actividad, lo que permite identificar las `rutas` necesarias para la realización de servicio. Dichas rutas se representan en el modelo de navegación, permitiendo de esta manera el desarrollo de sistemas más intuitivos y fácilmente navegables por el usuario.

Antes de la presentación del método para el modelado del hipertexto de MIDAS y los nuevos modelos propuestos, es necesario introducir un nuevo conjunto de conceptos asociados a la representación de servicios como elementos de modelado.

4.3.1 Meta-modelo de servicios

Un **`servicio`** es una funcionalidad, ofrecida por el sistema y con un resultado que satisface una necesidad especifica del usuario. Para la representación de servicios, el modelo mas apropiado es el modelo de casos de uso, sin embargo no siempre un caso de uso constituye un servicio esperado por el usuario. Un **`caso de uso`** se define como "la especificación de una secuencia de acciones realizadas por el sistema, con un resultado observable y que es de valor para uno o mas de los actores del sistema" [40]. Así, en el caso de un sistema de compra de billetes de avión, registrarse como cliente es un caso de uso, puesto tiene un resultado observable y de valor para el cliente, sin embargo, éste no es un servicio, puesto que no satisface una necesidad específica del usuario. En ese caso la necesidad del usuario es comprar un billete de avión y no registrarse como cliente.

La Figura 4.5 presenta el meta-modelo de servicios que identifica los diferentes tipos de servicios que intervienen en el método del modelado del hipertexto de MIDAS, así como su relación con el modelo de casos de uso.

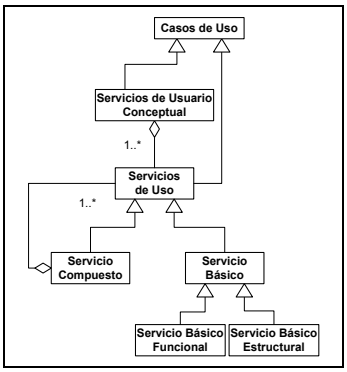


Fig. 4.5.- Meta-modelo de servicios en MIDAS

Los **servicios de usuario conceptuales** son un tipo especial de casos de uso y se utilizan para representar los servicios que un usuario requiere del sistema (v.g. comprar un billete de avión o reservar una habitación en un hotel).

Dichos servicios de usuario conceptuales se descomponen en servicios más sencillos que se ha denominado **servicios de uso**, y que son las funcionalidades requeridas por el SIW para completar dicho servicio de usuario conceptual. Como puede verse en la Figura 4.5, un servicio de usuario conceptual es una agregación de servicios de uso.

En base a la clasificación de servicios web propuesta en [45], los servicio de uso pueden ser a su vez servicios básicos o compuestos. Un **servicio de uso compuesto** es una agregación de varios servicios de uso, los cuales pueden ser básicos o compuestos, como se muestra en la Figura 4.5. Así, volviendo al caso de la compra del billete de avión y suponiendo que el sistema permitiera hacer primero la reserva del vuelo y luego pagar y obtener el billete, un ejemplo de servicio de uso compuesto sería 'reservar un vuelo', este servicio de uso compuesto se descompone en una serie de tareas a realizar como buscar el vuelo indicando origen, destino y fechas, ver vuelos disponibles, seleccionar horarios y precio y registrar datos personales. Un **servicio de uso básico** es una funcionalidad atómica del SIW desde el punto de vista del usuario (v.g. registrar datos personales, en el ejemplo ya mencionado).

Finalmente, un servicio de uso básico puede ser estructural o funcional como se muestra en la misma figura 4.5. Un servicio de uso básico es **estructural** cuando proporciona una vista de datos de los datos que conforman el SIW (v.g. ver vuelos disponibles). Un servicio de uso básico es **funcional** cuando implica una interacción con el usuario, generalmente cuando se requiere algún dato de entrada (v.g. buscar el vuelo indicando origen, destino y fechas o registrar los datos personales).

Estos nuevos conceptos, así como sus modelos asociados y el método para obtenerlos, se describirán en la siguiente sección. En la sección 4.3.2 se describe el método para el modelado del hipertexto de MIDAS y en las subsecciones 4.3.2.1, 4.3.2.2, 4.3.2.3 y 4.3.2.4 se presentan los distintos modelos propuestos. Finalmente, en la subsección 4.3.2.5 se describen las reglas necesarias para la transformación entre los modelos propuestos.

4.3.2 Modelo y método

El método para el modelado del hipertexto de MIDAS propone un proceso guiado por los servicios que el usuario espera de la aplicación, para la obtención del modelo navegacional de un SIW. El proceso, como se muestra en la Figura 4.6, toma como entradas el conjunto de requisitos de usuario así como el modelo de datos conceptual y tiene como salida un modelo de navegación extendido.

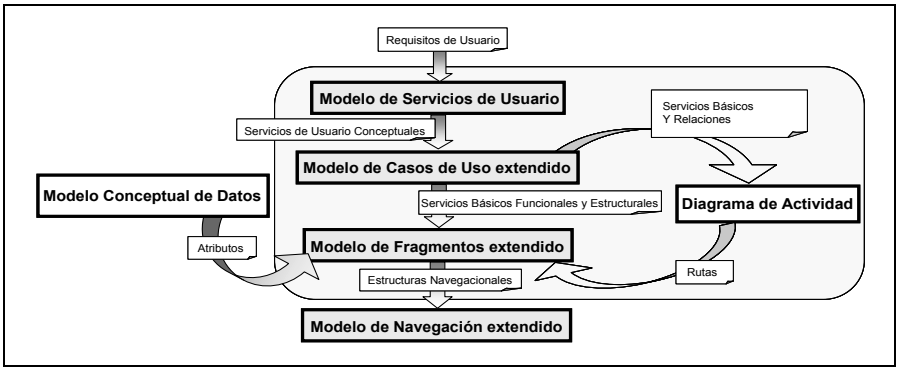


Fig. 4.6.- Proceso del método del modelado del hipertexto en MIDAS

Los nuevos modelos propuestos que aparecen sombreado en la Figura 4.6, son el `modelo de servicios de usuario`, el `modelo de casos de uso extendido`, el `modelo de fragmentos extendido` y el `modelo de navegación extendido`. Además, como se ha dicho, se propone el uso del diagrama de actividad para modelar proceso para la realización de cada servicio y establecer la ruta que guiará al usuario mientras navega en el SIW.

Junto con el proceso, se han definido un conjunto de nuevos modelos que se representa mediante extensiones UML, y las reglas de transformación entre los modelos. La Tabla 4.3 resume los modelos necesarios para el modelado del hipertexto, incluyendo la notación utilizada para cada uno de ellos.

Tabla 4.3. Modelado del hipertexto en MIDAS

Nivel	Tarea	Modelo	Notación
PIM	Modelado del Hipertexto	Modelo de Clases	Diagrama de clases de UML
		Modelo de Servicios de Usuario	Basado en Diagrama de Casos de Uso de UML + Extensiones de MIDAS
		Modelo de Casos de Uso Extendido	Basado en Diagrama de Casos de Uso de UML + Extensiones de MIDAS
		Modelo de Fragmentos extendido	Basado en Modelo de Fragmentos RMM + Extensiones de MIDAS
		Modelo de Navegación extendido	Basado en Modelo de Navegación UWE + Extensiones MIDAS

En las siguientes subsecciones se describen los nuevos modelos, explicando en cada caso que se modela en cada uno de ellos y cuáles son los estereotipos utilizados para la representación de los nuevos conceptos presentados en la sección anterior. La subsección 4.3.2.1 presenta el modelo de servicio de usuario. La subsección 4.3.2.2 presenta el modelo de caso de uso extendido. En la subsección 4.3.2.3 se presenta el modelo de fragmentos extendido, explicando brevemente que es un modelo de fragmentos y además se define el concepto de rutas de navegación. La subsección 4.3.2.4 presenta el modelo de navegación extendido. En la subsección 4.3.2.5 se resume el método para el modelado del hipertexto de MIDAS y se presentan las reglas para la transformación entre modelos.

4.3.2.1. Modelo de servicios de usuario

Este modelo es un modelo de casos de uso en el cual se representan los servicios que el usuario requiere del sistema, es decir los servicios de usuario conceptuales. Dichos servicios de usuario conceptuales se representan como casos de uso estereotipados con <<CUS>> (*Conceptual User Service*). La generación del modelo de servicios de usuario comienza con la identificación de los servicios ofrecidos a los usuarios, los cuales son obtenidos a través de los distintos métodos de captura de requisitos.

Para aclarar como se construye este modelo se presenta como ejemplo un SIW que ofrece distintos servicios a través de Internet para la realización de viajes. En este caso algunos de los servicios requeridos por los usuarios que desean viajar pueden ser: 'comprar un billete de avión', 'reservar una habitación en un hotel' o 'alquilar un coche'. En la Figura 4.7 se muestra el modelo de servicio de usuario para este ejemplo, en el cual se representa al usuario como un actor (Viajero) y como servicios de usuarios conceptuales, los servicios requeridos por dicho usuario.

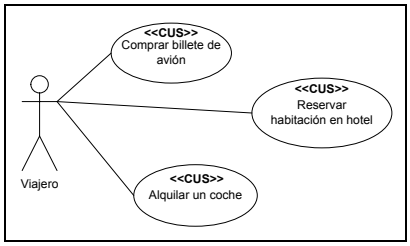


Fig. 4.7.- Modelo de servicios de usuario

Puede observarse que en el modelo de la Figura 4.7 sólo se representan los servicios que se ofrecerán a un tipo de usuario, los viajeros. Sin embargo, este tipo de sistema podrá ofrecer otros servicios a otros tipos usuarios (v.g. a los proveedores de servicios como hoteles o alquiler de coches). Para representar los servicios requeridos por cada tipo de usuario del sistema, se representan como actores cada uno de dichos usuario y asociado a éstos, los servicios requeridos por ellos.

4.3.2.2. Modelo de casos de uso extendido

Este modelo es también un modelo de caso de uso en el cual se representan las diferentes funcionalidades y tareas en que se descomponen los servicios de usuario conceptuales de la actividad anterior. Estas funcionalidades son los servicios de uso, los cuales pueden ser compuestos o básicos, éstos últimos pueden ser a su vez funcionales o estructurales.

La generación del modelo de casos de uso extendido comienza con la descomposición de los servicios de usuario conceptuales identificados en la actividad anterior, en las diferentes tareas o funcionalidades necesarias del SIW necesarios para llevar a cabo el servicio. A partir de esta descomposición, habrá que realizar el siguiente conjunto de actividades de forma iterativa:

a) identificar los servicios de uso como básicos y compuestos,

b) clasificar como estructurales o funcionales los servicios de uso básicos,

c) añadir las relaciones *include* y *extend* entre los servicios de uso.

Esta iteración termina cuando el modelo de casos de uso extendido contenga sólo servicios de uso básicos; es decir, si al descomponer los servicios de usuarios conceptuales se identifican servicios de usos compuestos, éstos se deben descomponer nuevamente realizando las actividades a, b y c hasta la obtención de servicios de usos básicos. En definitiva, el modelo de casos de uso extendido incluye: actores, relaciones *include* y *extend*, y los servicios de uso básicos estructurales y funcionales. Los actores y las relaciones *include* y *extend* tienen aquí la misma semántica que en el modelo de casos de uso. "Una relación *include* especifica la existencia de un flujo de eventos en el cual el caso de uso base incluye el comportamiento del otro" y "una relación *extend* especifica que el comportamiento de un caso de uso base puede ser opcionalmente extendido por el comportamiento de otro caso de uso" [40]. Los servicios de uso se representan como casos de uso, estereotipados de la siguiente forma: <<CS>> para representar un servicio de uso compuesto *(Composite Service)*, <<FBS>> para representar un servicio de uso básico funcional *(Functional Basic*

Service) y <<SBS>> para representar un servicio de uso básico estructural *(Structural Basic Service)*.

Volviendo al ejemplo presentado en la sección anterior de los servicios para la realización de viajes, el servicio de usuario conceptual 'comprar un billete de avión' se descompone en varios servicios de usos, los cuales incluyen: realizar la búsqueda del vuelo, seleccionar uno de los vuelos disponibles, registrar los datos personales del viajero, introducir los datos del envío del billete, en caso de que el billete no sea electrónico, y finalmente pagar el vuelo, introduciendo los datos de la tarjeta de crédito. Todos ellos son servicios de uso básicos. Esta descomposición se representa en el modelo de caso de uso extendido de la Figura 4.8(a).

La siguiente actividad consiste en el análisis de los servicios de uso básico para identificarlos como estructurales o funcionales. Este análisis se realiza desde el punto de vista de la presentación del SIW. Un servicio de uso básico es estructural cuando proporciona una vista de los datos que conforman el SIW. En el ejemplo, 'seleccionar vuelo entre los disponibles' es un servicio de uso básico estructural, puesto que lo que el usuario ve, es una listado de los vuelos entre los cuales deberá escoger uno. Un servicio de uso básico es funcional cuando implica una interacción con el usuario, generalmente cuando se requiere algún dato de entrada. En el ejemplo, 'buscar vuelo', 'registrar datos personales', 'registrar datos del envío' y 'pagar vuelo', son servicios de uso básicos funcionales.

Como última actividad, hay que identificar las relaciones *include* y *extend* entre los servicios de uso identificados. En el ejemplo, el servicio de uso básico 'seleccionar vuelo entre los disponibles' está asociado con el servicio de uso básico 'buscar vuelos' por una relación *incluye*; puesto que para la selección, primero se deberá buscar el vuelo indicando los datos requeridos (v.g. origen y destino, fechas). Además, el servicio de uso básico 'registrar datos personales' está asociado con el servicio de uso básico 'registrar datos del envío' por una relación *extend*; puesto que el segundo representa una funcionalidad opcional que se realizará en caso de que el viajero requiera la impresión del billete. Finalmente, los servicios de uso básico 'seleccionar vuelos entre los disponibles' y 'registrar datos personales', están asociados al servicio de uso básico 'pagar vuelo' por una relación *include*.

La Figura 4.8(a) muestra el modelo de casos de uso extendido obtenido en el que se representa el servicio de usuario conceptual 'comprar billete de avión' de la Figura 4.7 y en Figura 4.8(b) se muestra el diagrama de actividad asociado a dicho servicio. El diagrama de actividad se realiza representando como actividades cada uno de los servicios de uso básicos, tanto funcionales como estructurales, e identificando la secuencia en la cual deben ser realizados. Así, a través del diagrama de actividad es posible representar el proceso para la ejecución de cada servicio, lo cual permite posteriormente identificar las rutas necesarias para la realización de servicio. En la siguiente subsección se define el concepto de ruta y sus implicaciones en el modelo de navegación.

El mismo procedimiento aplicado al servicio de usuario conceptual, comprar billete de avión, se aplica a los servicios de usuario conceptuales 'alquilar un coche' y 'reservar habitación en hotel'.

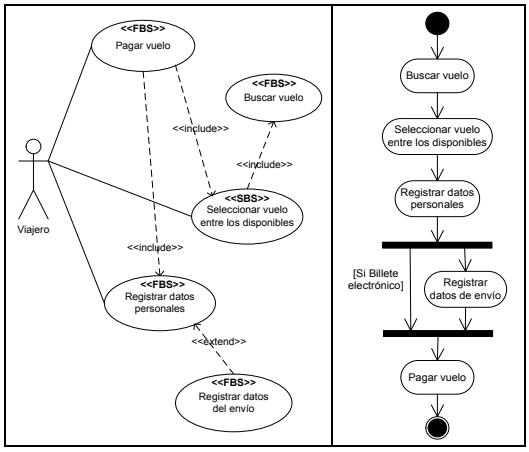


Fig. 4.8.- (a) Modelo de caso de uso extendido - (b) Diagrama de Actividad

4.3.2.3. Modelo de fragmentos extendido: Rutas de navegación

El **`modelo de fragmentos`** se define como la descomposición del sistema en unidades significativas, denominadas fragmentos, y los hiperenlaces entre dichos fragmentos [30]. Este modelo se representa a través de un diagrama de clases, en el cual los fragmentos son representados como clases estereotipadas y los hiperenlaces como relaciones entre clases. En dicho modelo cada fragmento, representa una unidad de información que se va a mostrar en la web de forma agrupada (v.g el fragmento, buscar vuelo en la Figura 4.9, representa la información que se mostrará cuando el usuario desee realizar esta actividad). En el modelo de fragmentos extendido se identifican dos tipos de fragmentos: estructurales y funcionales. Un **`fragmento estructural`** es un fragmento tal y como se entiende en el modelo de fragmentos mencionado anteriormente. Un **`fragmento funcional`** es un fragmento en el que, además de la información que será mostrada en la web, pueden aparecen otro tipo de información (v.g. datos que el usuario deberá introducir) o funcionalidades; así, un fragmento funcional permite la representación de la interacción del usuario con el SIW. Los diferentes fragmentos se estereotipan con <<SS>> y <<FS>> para representar respectivamente los fragmentos estructurales *(Structural Slice)* y fragmentos funcionales *(Functional Slice)*.

La generación del modelo de fragmentos extendido comienza con `la identificación de los fragmentos`. Cada servicio básico estructural y funcional del modelo de casos de uso extendido se transformará en fragmentos estructurales y funcionales respectivamente. Así en el ejemplo, 'seleccionar vuelo entre los disponibles' es un fragmento estructural y 'buscar vuelo', 'registrar datos personales', 'registrar datos del envío' y 'pagar vuelo', son fragmentos funcionales (ver Figura 4.9).

Una vez identificados los fragmentos funcionales y estructurales, y para continuar con la generación del modelo de fragmentos extendido, hay que realizar la `identificación de los atributos de los fragmentos`, dichos

atributos se obtienen del modelo de datos conceptual. En el ejemplo, dicho modelo está formado por las dos clases: Vuelo y Viajero. Así, los fragmentos 'buscar vuelo' y 'seleccionar vuelo entre los disponibles', tomarán los atributos de la clase Vuelo y los fragmentos, 'registrar datos personales' y 'pagar vuelo', tomarán los atributos de la clase Viajero.

La última actividad a realizar para completar el modelo de fragmentos extendido es `asociar a cada servicio de usuario conceptual su ruta específica`. Para explicar exactamente qué indica la ruta y por qué se cree que es necesario identificarla en el modelo de navegación de un SIW se va a poner un ejemplo. Cuando se realiza un viaje en coche a un lugar desconocido, es importante tener un mapa o una guía que indique la ruta a seguir. Así, con la ruta marcada, es posible saber exactamente que camino tomar en cada momento, evitando perderse antes de llegar al destino. De la misma manera, cuando un usuario accede a un SIW por primera vez, generalmente sólo sabrá qué es lo que quiere hacer, qué servicios desea, pero seguramente no sabrá como hacerlo puesto que, a diferencia de los sistemas de información tradicionales, el usuario de un SIW no dispone de un manual de usuario le indique como usarlo. Así, realizando una analogía con el ejemplo del viaje, el hecho de marcar la ruta que el usuario deberá seguir para la realización de cada servicio en el mapa navegacional del SIW, le ayudará a saber en cada momento qué hiperenlace debe tomar, cual es la próxima actividad a realizar, cuántas actividades le quedan por realizar, etc.; evitando así que el usuario se pierda en el espacio de navegación. De esta manera, el SIW obtenido será mucho más intuitivo y amigable para el usuario.

En el método del modelado del hipertexto de MIDAS, se propone definir una ruta para cada servicio de usuario conceptual. Esta ruta representa la secuencia de pasos que el usuario debe seguir para completar el servicio y se haya representada a través del diagrama de actividad asociado a dicho servicio de usuario conceptual. En el modelo de fragmentos extendido, los fragmentos se enlazan con una relación dirigida y siguiendo la ruta. Cada ruta debe quedar identificada en el modelo de fragmentos extendido, con el estereotipo <<route>> y el valor etiquetado {NombreRuta}. Una ruta puede además tener caminos alternativos o subrutas. En ese caso, si la ruta `X` tiene la subruta `Y`, dicha subruta se estereotipa como <<route>> y el valor etiquetado {`X.Y`}.

En el ejemplo, el diagrama de actividad asociado al servicio 'comprar billete de avión', que se muestra en la Figura 4.8(b), indica que el viajero primero debe buscar el vuelo, y luego seleccionar uno de los disponibles. Una vez elegido el vuelo deberá registrar sus datos personales y luego pagar. Esta ruta ha quedado identificada como <<route>> {CB} y se representa en el modelo de fragmentos extendido de la Figura 4.9. Además, se representa la subruta 'registrar datos de envío', que el usuario deberá tomar en caso de que el billete no sea electrónico, quedando identificado en la figura como <<route>> {CB.DE}.

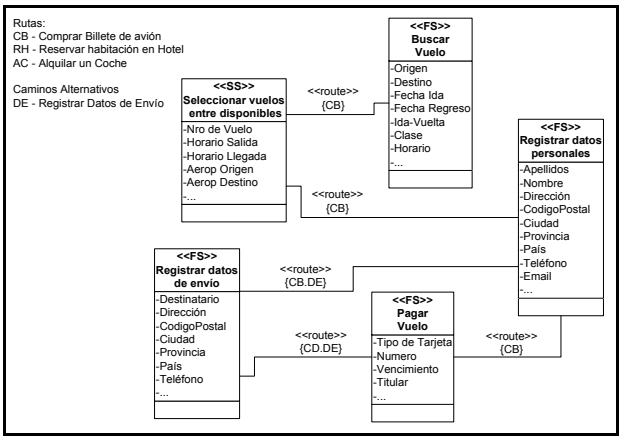


Fig. 4.9.- Modelo de fragmentos extendido

4.3.2.4. Modelo de navegación extendido

El modelo de navegación se obtiene generalmente añadiendo las `estructuras de navegación` (v.g. menús, índices) al modelo de fragmentos. De la misma manera, el modelo de navegación extendido es el modelo de fragmentos extendido más las estructuras de navegación, que en este caso concreto se obtiene de la extensión UML propuesta en UWE [28]. De este modo el modelo de navegación extendido incorpora los fragmentos estructurales y funcionales así como las rutas específicas del modelo que le precede, así como las estructuras concretas de navegación (menús, índices, etc.).

La generación del modelo de navegación extendido comienza por `identificar las estructuras de navegación`. El método del modelado del hipertexto de MIDAS propone introducir un menú principal en el que se represente una entrada por cada servicio de usuario conceptual identificado en el modelo de servicios de usuario, que supondrá el inicio de cada una de las rutas identificadas en el modelo de fragmentos extendido. Cada ruta, que en este modelo también debe quedar identificada, debe ser tenida en cuenta en la implementación, es decir, las paginas web del SIW obtenidas a través de las clases del modelo navegacional estereotipadas con <<FS>> y <<SS>> deberán mostrar la ruta del servicio que el usuario está realizando. De esta manera el SIW estará preparado para que el usuario realice cada uno de los servicios que desea, siguiendo la ruta adecuada y sin perderse durante la navegación.

En la Figura 4.10 se muestra el modelo de navegación extendido para el ejemplo expuesto. El modelo presenta un menú principal con una entrada para cada servicio de usuario conceptual: 'comprar billete de avión', 'reservar habitación en hotel' y 'alquilar un coche'. Además, se identifican las rutas para cada servicio. Si el usuario elige la opción, 'comprar billete de avión', el sistema le guiará en las actividades a realizar, así primero deberá realizar la búsqueda, luego seleccionar el vuelo, registrar sus datos, etc. De esta manera, el usuario podrá llevar a cabo su objetivo, es decir lo que espera del sistema, de una manera intuitiva.

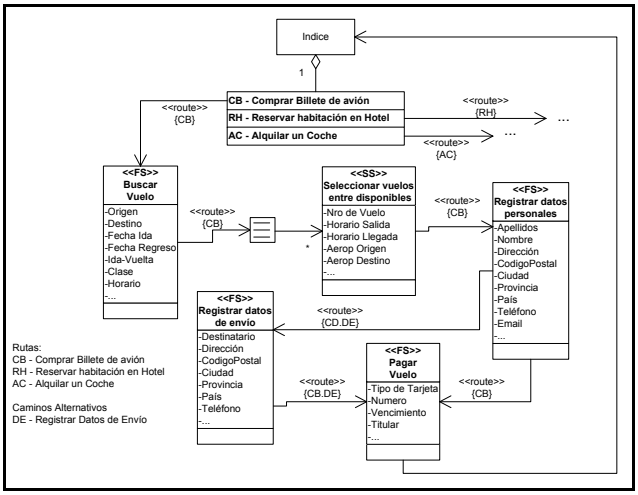


Fig. 4.10.- Modelo de navegación extendido

4.3.2.5. Reglas de transformación entre modelos

El método para el modelado del hipertexto de MIDAS presenta un proceso que finaliza con la obtención del modelo de navegación extendido. Dicho proceso (Figura 4.6) incluye cuatro nuevos modelos los cuales han sido explicados en subsecciones anteriores. En la Tabla 4.4 se presentan las guías para la transformación entre los modelos propuestos, todos ellos a nivel PIM. Dicha tabla presenta por cada modelo propuesto, los modelos o actividades de entrada y las transformaciones necesarias en cada uno de ellos para la generación del modelo en cuestión.

Tabla 4.4. Guías para la transformación entre modelos PIM propuestos por MIDAS.

Modelos propuestos por MIDAS	PIM Entrada	Transformación
Modelo de Servicio de Usuario	Este modelo se obtiene a partir de los requisitos de usuarios. Los servicios requeridos por el usuario son representados como <<CUS>>	
Modelo de Casos de Uso extendido	Modelo de Servicio de Usuario	Cada CUS (servicio de usuario conceptual) se descompone en servicios de usuario, los cuales pueden ser: compuestos (CS), básicos estructurales (SBS) y básicos funcionales (FBS).
Modelo de Fragmentos extendido	Modelo de Casos de Uso extendido	Cada FBS es transformado en un fragmento funcional (FS) y cada SBS es transformado en un fragmento estructural (SS).
	Diagrama de Actividad	El flujo de trabajo asociado a cada CUS mostrado en este modelo, se representa como hiperenlaces dirigidos entre los distintos fragmentos que forman parte del CUS, identificando así la ruta para la realización del mismo. Las rutas y subrutas son luego estereotipadas.
	Modelo Conceptual de Datos	Los atributos de las clases de este modelo, son distribuidos en los distintos fragmentos según los datos que sean necesarios mostrar o ingresar en cada fragmento estructural o funcional.
Modelo de Navegación extendido	Modelo de Fragmentos extendido	Se agregan las estructuras de navegación como índices, menú, etc.

4.4 Desarrollo de la Base de Datos en MIDAS

Esta sección se centra en el aspecto de `contenido` de MIDAS, que se corresponde con el concepto tradicional de una BD. El desarrollo de una BD depende de varios aspectos: en primer lugar, de si ya existe una BD en la organización y, por otro lado, de la tecnología que se desea utilizar, es decir, si se desea utilizar una Base de Datos (Objeto-) Relacional (BDOR) o una BD XML. MIDAS define un marco genérico y un proceso específico para el desarrollo de una BD web que tiene en cuenta estos aspectos. Una descripción detallada del proceso completo de MIDAS, sus tareas, modelos y notaciones se puede encontrar en [14,36].

Existen diferentes formas de desarrollar una BD dependiendo de la tecnología usada:

- A nivel de `PIM` se realiza el diseño conceptual de datos utilizando el modelo conceptual de datos, sin tener en cuenta la tecnología seleccionada, ya que se trata de un modelo independiente de la plataforma. Este PIM de datos se representa mediante un diagrama de clases UML.
- A nivel de `PSM` se lleva a cabo, por un lado, el diseño lógico de datos, partiendo del PIM de datos obtenido y, por otro lado, la implementación de la BD. Sin embargo, ahora sí que habrá que tener en cuenta la tecnología seleccionada:
    - **Tecnología Objeto-Relacional:** En el diseño lógico de la BD se obtiene, a nivel de PSM, el modelo lógico de datos estándar en

SQL:2003 y el específico en el producto seleccionado (v.g. Oracle 9*i*) aplicando las reglas de transformación que se resumen en la subsección 4.4.1.3. Para la implementación de la BD se usa el SQL del producto seleccionado.

- **Tecnología XML:** En la primera tarea, es decir, el diseño lógico de la BD, se parte del PIM de datos para obtener a nivel de PSM el esquema XML aplicando las reglas de transformación que se resumen en la subsección 4.4.1.3. Para la implementación se usa una BD XML para almacenar la información XML conforme al esquema XML obtenido.

Para estos modelos se utiliza UML como notación, para lo que se ha definido una extensión para el diseño de BDOR [35,37] y otra para representar esquemas XML [51,52], que se resumen en las subsecciones 4.4.1.1 y 4.4.1.2, respectivamente, de este capítulo.

La Tabla 4.5 resume las tareas, técnicas y notaciones que se deben llevar a cabo a la hora de desarrollar una BD web.

Tabla 4.5. Desarrollo de una BD web

Nivel	Tarea	Modelo		Notación
PIM	Diseño Conceptual de Datos	Modelo Conceptual de Datos		Diagrama de Clases (UML)
PSM	Diseño Lógico de Datos	Modelo Lógico de Datos	Objeto-Relacional	SQL:2003 (MIDAS-UML) Producto (MIDAS-UML)
			XML Schema	XML Schema (MIDAS-UML)
	Implementación de la BD	Objeto-Relacional		Código SQL
		XML Schema		Código BD XML

4.4.1. Modelos y método

Como ya se ha dicho, MIDAS propone usar UML como notación única para modelar todo el sistema. Sin embargo, UML no tiene una notación específica que permita representar todos los modelos necesarios para el desarrollo de un SIW; en concreto, no tiene una notación específica para representar un esquema de una BDOR o un esquema de una BD XML (XML Schema).

Para poder utilizar UML como lenguaje estándar para el modelado de BD web, es necesario extenderlo, de tal forma que permita representar tanto un esquema objeto-relacional como un esquema XML. De este modo, MIDAS define extensiones a UML para el modelado de BDOR, en concreto para el modelo del estándar SQL:2003 y del producto Oracle9*i*, y para el modelado de BD XML, concretamente para poder representar con UML esquemas XML. En las siguientes subsecciones 4.4.1.1 y 4.4.1.2 se verá un resumen de estas extensiones.

4.4.1.1 Extensión UML para BD Objeto-Relacionales

Tanto a nivel PIM como a nivel PSM, se propone representar los modelos con UML. En el caso de BDOR, se representarán dos PSMs, uno de más alto nivel basado en el estándar para BDOR, SQL:2003, y otro de más bajo nivel que corresponde al modelo concreto del producto seleccionado para la implementación que, en este caso, es Oracle9*i*. Para la representación de un modelo puramente relacional, puede utilizarse la extensión de UML propuesta por Ambler [4]. Para la representación del PSM de alto nivel para el estándar SQL:2003, así como del PSM específico para Oracle 9*i*, MIDAS introduce nuevas extensiones que pueden verse detalladas en [35].

Estas extensiones definen un conjunto de estereotipos, valores etiquetados y restricciones que permiten representar en notación gráfica en UML cada elemento del modelo OR del SQL: 2003 y del producto Oracle9*i*.

La figura 4.11 resume la propuesta de extensión de UML para el estándar SQL:2003 mediante el meta-modelo correspondiente; por motivos de espacio y por ser muy similar a este, no se incluye el meta-modelo para el producto seleccionado.

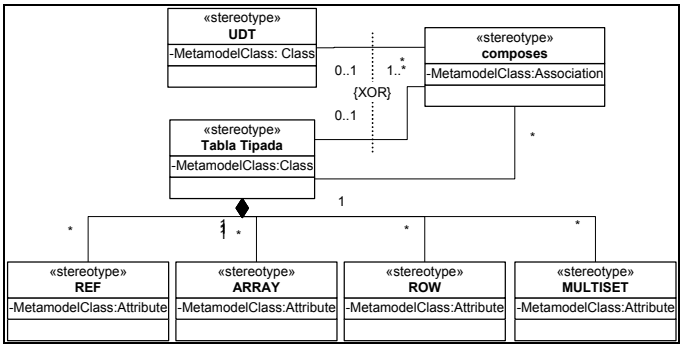


Fig. 4.11.- Meta-modelo de la extensión de UML para SQL:2003

A la hora de elegir los estereotipos se ha seguido el siguiente criterio:

Para SQL:2003:

- Se han considerado los `tipos estructurados` (**`<<udt>>`**: tipos definidos por el usuario) y las `tablas tipadas` como clases estereotipadas porque son definidas explícitamente en el esquema en SQL. Las tablas tipadas se definen como clases estereotipadas con **`<<persistent>>`** y su definición implica la definición de un tipo estructurado `<<udt>>`, que será el tipo de la tabla, con su correspondiente extensión.
- Los tipos REF, ARRAY, ROW y MULTISET se consideran como atributos estereotipados con **`<<ref>>`**, **`<<array>>`**, **`<<row>>`** y **`<<multiset>>`,** respectivamente.
- Una asociación **`<<composes>>`** es un tipo de asociación especial que une un tipo de datos definido por el usuario `<<udt>>` con la clase que lo utiliza. Es una interrelación unidireccional, representando la dirección de la asociación por medio de una flecha que parte del tipo definido por el usuario y llega a la clase que utiliza dicho tipo. Únicamente puede utilizarse para unir

una clase <<persistent>> con una clase <<udt>>. Además, si una clase tiene atributos de tipo referencia o de tipo colección, cuyos elementos sean instancias, bien de una clase <<persistent>>, o bien referencias a una clase <<persistent>>, también existiría una asociación <<composes>> entre la clase <<persistent>> referenciada y la que contiene el atributo de tipo referencia o colección.

Para Oracle9i:

- Se han considerado tipos de objetos, tablas tipadas y tablas anidadas (*nested tables*) como clases estereotipadas por el mismo motivo que en el caso del SQL:2003. Cada clase <<persistent>> en Oracle9i se corresponde con un tipo de objeto con su correspondiente extensión, que es la tabla de tipo de objetos.
- El tipo REF se ha considerado como un atributo estereotipado con <<ref>>, porque no puede ser definido explícitamente en el esquema en SQL.
- Un tipo ARRAY representa un tipo colección indexado y limitado y puede, o no, ser definido explícitamente en el esquema, por lo que se permitirán las dos posibilidades: definirlo como un atributo estereotipado o como una clase estereotipada con <<array>>. Se utilizará la clase estereotipada cuando el tipo ARRAY se haya definido explícitamente en el esquema.
- Un tipo NESTED TABLE (tabla anidada) representa un tipo colección no indexado ni limitado y se define como una clase estereotipada con **<<nt>>**.
- Una asociación <<composes>> es un tipo especial de interrelación que une un tipo <<udt>>, <<array>> o <<nt>> con la clase que lo utiliza. Es una interrelación unidireccional, donde la dirección de la asociación se representa mediante una flecha que apunta a la clase que utiliza el tipo de datos. De la misma forma que en el estándar todo atributo de tipo <<ref>> o de tipo colección, cuyos elementos sean instancias de una clase <<persistent>> o de tipo <<ref>> a una clase <<persistent>>, también implica una asociación con otra clase.

4.4.1.2 Extensión UML para BD XML

En el caso de seleccionar la tecnología XML para el PSM de datos en MIDAS se propone usar el modelo de esquemas XML (XML Schemas [56]), representado también mediante UML extendido. La Figura 4.12 muestra el meta-modelo de la extensión de UML para representar esquemas XML; para más detalle sobre la misma ver [51].

La extensión define un conjunto de estereotipos, valores etiquetados y restricciones que permiten representar en notación gráfica en UML todos los componentes de un esquema XML, manteniendo las asociaciones, el orden y el anidamiento entre los distintos elementos.

A la hora de elegir los estereotipos se ha seguido el siguiente criterio:

- Los elementos **ELEMENT** se han considerado clases estereotipadas con <<ELEMENT>> porque están explícitamente definidas en el XML Schema. Los atributos de un ELEMENT se han considerado atributos estereotipados de las clases que representan al ELEMENT.

- Los tipos **`complexType`** se han considerado clases estereotipadas con <<complexType>> si tienen nombre. En este caso, el tipo `complexType` estará relacionado con el tipo `ELEMENT` o el tipo que lo use mediante una asociación **`<<uses>>`**. Si no tiene nombre, se representará de forma implícita mediante el **`compositor`** que compone.
- Los `compositors` se consideran composiciones, que son un tipo especial de asociación, estereotipadas. Sus estereotipos dependerán del tipo de compositor: **`<<Choice>>`**, **`<<Sequence>>`** o **`<<All>>`**.
- Los tipos **`simpleType`** se han considerado clases estereotipadas con el mismo nombre que el elemento que los contiene. Estará relacionado con su padre (`ELEMENT`) mediante una composición estereotipada con **`<<simpleType>>`**.
- Los tipos **`complexContent`** se han considerado como clases estereotipadas que deben estar relacionadas con una relación de herencia con el tipo padre, que debe ser un `complexType`, que se está redefiniendo mediante el tipo `complexContent`.
- Los tipos **`simpleContent`** se han considerado clases estereotipadas que están relacionadas mediante una relación de herencia con el tipo padre (`simpleType` o `complexType`) que redefine el tipo `simpleContent`.
- Para cada elemento, tipo y atributo se debe especificar al lado del nombre de la clase, tipo o atributo el número de orden, incluyendo como un prefijo el número de orden de elemento o tipo padre.

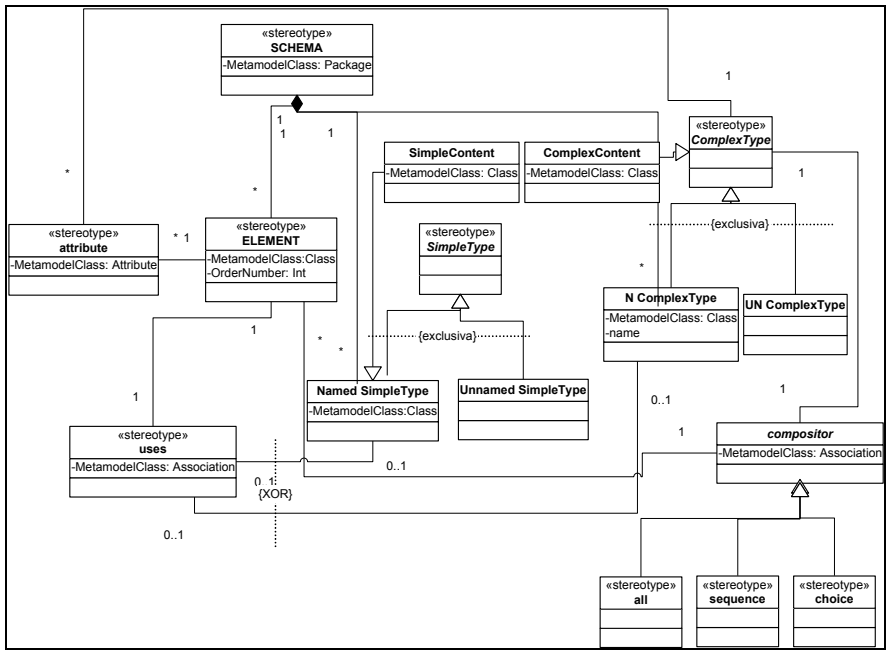


Fig. 4.12.- Metamodelo de la extensión de UML para esquemas XML

4.4.1.3 Reglas de transformación entre modelos

De forma similar a como las metodologías para bases de datos relacionales proponen algunas reglas para la transformación de un esquema conceptual en un esquema lógico estándar, en MIDAS se proponen reglas de transformación para pasar del nivel de PIM al de PSM. Esta técnica sugiere algunas reglas que deberían tenerse en cuenta únicamente como guías.

La Tabla 4.6 resume las reglas para transformación de un PIM a un PSM utilizando tecnología (objeto-)relacional; se incluyen tanto las transformaciones a realizar para obtener el PSM de alto nivel (SQL:2003) como para el PSM de bajo nivel (se ha considerado como producto Oracle 9*i)*. Estas reglas pueden encontrarse detalladas en [37].

Tabla 4.6. Reglas de transformación para pasar del PIM al PSM de datos: BDOR

PIM de datos	**PSM de datos**	
	SQL:2003	**Oracle9i**
Clase	Tipo Estructurado	Tipo de Objeto
Extensión de la Clase	Tabla definida sobre el Tipo Estructurado	Tabla definida sobre Tipo de Objeto
Atributo	Atributo	Atributo
Multivaluado	ARRAY	VARRAY
Compuesto	ROW / Tipo Estructurado en columna	Tipo de Objeto en columna
Calculado/Derivado	Disparador/Método	Disparador/Método
Asociación		
Uno-a-Uno	REF/REF	REF/REF
Uno-a-Muchos	REF/ARRAY	REF/Nested Table
Muchos-a- Muchos	ARRAY/ARRAY	Nested Table/Nested Table
Agregación	ARRAY	Nested Table
Generalización	Tipos/Tablas Tipadas	Subtipos

En la Tabla 4.7 se resumen las reglas de transformación para pasar del PIM de datos al correspondiente PSM utilizando tecnología de bases de datos XML. En [52] aparecen las reglas de forma detallada.

Tabla 4.7. Reglas de transformación para pasar del PIM al PSM de datos: BD XML

PIM de datos	**PSM de datos**
PIM de datos	Esquema XML
Clase	Elemento XML
Subclase (generalización)	Tipo complejo (*complexType*)
Parte (composición)	Tipo complejo (*complexType*)
Atributo	Subelementos
Obligatorio	*minOccurs* =1 (valor por defecto)
Opcional	*minOccurs* = 0
Multivaluado	*maxOccurs*= N
Compuesto	Tipo complejo (*all* | *sequence*)
Selección (choice)	Tipo complejo *choice*
Asociación	
1:1	Subelemento (de cualquier elemento) para la asociación, con un tipo complejo que incluya un elemento de tipo REF
1:N	Subelemento (cardinalidad N) para asociación, con un tipo complejo que incluya un elemento de tipo REF. (cardinalidad 1)
N:M	Subelemento (de cualquier elemento) para la asociación con un tipo complejo de tipo secuencia que incluya elementos de tipo REF
Agregación	
Cardinalidad Mínima: 1	Subelemento con un tipo complejo con un elemento REF
Cardinalidad Máxima: N	Subelemento con un tipo complejo de tipo secuencia de elementos REF.
Composición	Composición de tipo *all* relacionando el compositor con las partes
Generalización	Tipo complejo de tipo *choice*

4.5 MIDAS: Lecciones aprendidas

Para validar MIDAS, es necesario comprobar distintos aspectos:

a) Validar que cada una de las extensiones propuestas permiten representar cada uno de los constructores de los modelos para los que fueron definidos.
b) Validar que las reglas de transformación propuestas permiten obtener, a partir de un modelo válido, otro modelo válido.
c) Validar que el modelo de proceso propuesto es aplicable y permite el desarrollo de un SIW aplicando técnicas de procesos ágiles en un desarrollo dirigido por modelos.
d) Finalmente, y una vez validados cada modelo o técnica por separado, es necesario probar la metodología completa para validar su aplicabilidad en casos reales.

Para realizar las validaciones a), b) y c) se han llevado a cabo diferentes casos de estudio. Los casos de estudio utilizados se han planteado, en general, como casos académicos definidos explícitamente para poder comprobar la validez de las extensiones y reglas definidas. Sin embargo, para la validación d) se están utilizando desarrollos de SIW reales que están permitiendo refinar la metodología.

A continuación, se resumen las principales lecciones aprendidas en dos casos académicos y en un desarrollo real. Los casos académicos que se presentan han permitido validar, por una parte, la integración de técnicas de desarrollo ágiles en una arquitectura dirigida por modelos y, por otra parte, el método de desarrollo del hipertexto. El caso real está permitiendo validar de manera más general la

aplicabilidad de MIDAS y, hasta el momento, ha sido especialmente útil para la validación del método de desarrollo de la BD.

4.5.1. Caso de estudio: experimento de integración de técnicas ágiles en una arquitectura dirigida por modelos

Se realizó un experimento con estudiantes de último curso de la titulación de Ingeniería Informática de la Universidad Rey Juan Carlos. Se planteó el desarrollo de una aplicación web basada en la información docente de una asignatura y los profesores involucrados en ella, con el fin de validar el proceso ágil de desarrollo dirigido por modelos (AMDD) de MIDAS. El proyecto se organizó con el personal dividido en ocho equipos de entre cuatro a diez estudiantes cada uno de ellos. Cada equipo fue dividido en pares, y todos los pares de un mismo equipo desarrollaron las mismas historias de usuario. Sin embargo, cada par realizaba su propia planificación de tareas, iteraciones e implementación y ejecutaba el conjunto de pruebas unitarias a su propio software. Además, cada par realizaba las pruebas del software generado por otro par, garantizando así la realimentación entre pares. De esta forma se intentaba fomentar el sentimiento de propiedad colectiva del código. Se les propuso que todo producto software generado estuviese disponible en una dirección web, habilitada para ese fin, con el fin de que tuviesen el compromiso de terminar en plazo y de que todo producto publicado en dicha dirección dispusiese de una calidad mínima garantizada. En cuanto al tiempo disponible para el desarrollo, éste estaba limitado a cuatro semanas.

Tras realizar la experiencia, se realizó un cuestionario a los estudiantes con el objetivo de recabar diferentes opiniones respecto al proceso de desarrollo de MIDAS. Algunas de las opiniones más destacadas por los alumnos son:

- `Separación de aspectos`: ha resultado de gran utilidad a los equipos puesto que ha facilitado la agrupación de requisitos por aspectos así como la planificación de versiones y de iteraciones y la priorización de los requisitos propiamente dichos.
- `Desarrollo por pares`: se ha considerado muy ventajoso el hecho de que desde los inicios del proyecto pudiera trabajarse de forma conjunta (incluidos el desarrollo de los modelos). Aunque los alumnos también han manifestado que, a pesar de ser de gran ayuda en los momentos iniciales, durante ciertas etapas ha representado ciertos problemas debido a opiniones opuestas, lo que ha representado retrasos en el desarrollo.
- `Pruebas`: el hecho de realizar pruebas entre cruzadas entre diferentes pares de desarrolladores ha garantizado la ejecución de las mismas y el conocimiento de las anomalías y problemas cuando éstos se producen, lo que mejora el sentimiento de propiedad colectiva del código.
- `Uso de estándares`: el uso de estándares de desarrollo ha facilitado en gran medida la comunicación entre los diferentes miembros del equipo, lo que supone sin duda una gran ventaja sobre otros tipos de métodos.
- `Cliente en el equipo`: el hecho de tener al cliente siempre disponible a través de la web, facilita en gran medida el trabajo de resolución de dudas y consultas acerca del desarrollo. Y dado que no siempre será posible tener un cliente en el equipo de desarrollo, esta alternativa se menciona como especialmente interesante.

En general, las lecciones aprendidas tras el análisis de los cuestionarios proceden de la mención de utilidad de las guías de transformación entre los modelos, puesto que aunque resulte costoso aprenderlas, facilita el trabajo a lo largo del proceso de desarrollo. Por ello, se realizó un esfuerzo en la definición de las reglas de transformación de forma que resultase más sencillo su aprendizaje. Los alumnos también destacaron la utilidad del uso de modelos con notación estándar, puesto que simplificaba la comunicación entre todos. Por este motivo, los modelos nuevos que se propusieron posteriormente en MIDAS se basaron en modelos UML o extensiones UML ya existentes.

4.5.2. Caso de estudio: Sistema para la administración de artículos enviados a un congreso

Para la validación del método para el desarrollo del hipertexto de MIDAS se analizó la herramienta Confmaster [19] que permite la organización de conferencias. A partir de dicha herramienta, se obtuvo el modelo de casos de uso del sistema y se aplicó una primera versión del método para el modelado del hipertexto propuesto por MIDAS [33]. Una vez analizado el modelo de navegación obtenido, se decidió refinar el método, proponiendo así el nuevo enfoque orientado a servicios de usuario y se representó el mismo sistema pero partiendo del modelo de servicios de usuario, es decir, identificando en primer lugar los servicios que el sistema debía ofrecer a los usuarios. Una vez obtenidos ambos modelos de navegación, el correspondiente a la aplicación original y el generado tras aplicar el método propuesto por MIDAS, éstos se compararon. Esta tarea permitió identificar las ventajas del modelo de navegación obtenido tras la aplicación del método para el modelado del hipertexto. Al partir desde una perspectiva orientada a servicios de usuarios, dicho modelo de navegación presenta un menú principal con una entrada por cada servicio conceptual de usuario, teniendo cada uno de estos servicios asociado una ruta específica que ayuda al usuario a navegar a través del SIW. Estas características del modelo de navegación obtenido permite la construcción de SIWs más intuitivos y más amigables para el usuario.

4.5.3. Desarrollo de un sistema real: SIW para la Gestión de Imágenes Médicas

El grupo Kybele y el grupo de Tecnología Electrónica, Bioingeniería e Imagen Médica trabajan en un proyecto conjunto con investigadores en neurociencias, cuyo objetivo es la implementación de un SIW para la gestión de imágenes médicas. Los objetivos de dicho sistema son ofrecer a los investigadores en neurociencias y médicos clínicos un sistema de almacenamiento de imágenes masivo de fácil acceso y así como el procesamiento y análisis adecuado y normalizado de las imágenes médicas, todo ello a través de la web. Además, todos los resultados obtenidos del procesamiento de las imágenes son almacenados, junto con las imágenes originales, en la misma BD. Esta BD de imágenes funcionales y sus resultados de procesamiento, servirán de archivo histórico que podrá ser consultado y utilizado a través de la web.

La BD tiene dos tipos de información: la información para la gestión de los estudios científicos destinados a la investigación en neurociencias que es altamente estructurada, y la información asociada a las imágenes médicas, con contenido semi-

estructurado y que se almacena en los dos formatos estándar soportados por el SIW: el formato DICOM (*Digital Image and COmmunications in Medicine*) [1] y el formato Analyze [38]. Por ello, se ha optado por implementar la información semi-estructurada en una BD XML y la parte estructurada en una BDOR, siguiendo en ambos casos el método propuesto en este trabajo. El proceso de desarrollo de esta BD (OR y XML) incluyó en primer lugar el modelado a nivel de PIM y posteriormente, el PSM de datos que se obtuvo tras aplicar las reglas de transformación propuestas específicas para ambas tecnologías.

Actualmente se está trabajando en el desarrollo del hipertexto del SIW y para ello se está utilizando la metodología propuesta por MIDAS. Se ha desarrollado el modelo de negocio del sistema y se han identificado las necesidades de los usuarios, que son investigadores en neurociencias y médicos clínicos. De esta forma, se han obtenido los servicios que el sistema deberá ofrecer. Dichos servicios son:

a) `Servicio de almacenamiento de datos`: la BD almacenará imágenes proporcionadas por los investigadores y resultados del procesamiento de las imágenes. Los investigadores y clínicos podrán tener acceso a estos datos y consultar por tipo de estudio, por patología, por modalidad, etc. con propósitos de diagnóstico, de investigación y de enseñanza.
b) `Servicio de Procesamiento`: se realizarán dos clases de procesamiento, análisis y segmentación.
c) `Servicio de visualización`: el proceso de la visualización consiste en reconstrucciones de imágenes (v.g. construir animaciones o imágenes tridimensionales a partir de varias imágenes en dos dimensiones) y además funciones multimodalidad (v.g. componer en una imagen anatómica y una imagen funcional).

El desarrollo de este sistema, hasta el momento, ha sido especialmente útil para la validación del método de desarrollo de BD, ya que ha permitido trabajar con un conjunto de datos estructurados y semi-estructurados. Para ello, se ha utilizado un repositorio común que integra facilidades de gestión OR y XML, en concreto se ha implementado en Oracle 9*i* [44]. La BD está actualmente en explotación y los resultados son satisfactorios para los usuarios de la misma. En la actualidad, se está trabajando en el desarrollo de los servicios de procesamiento y visualización que permitirán validar las propuestas de desarrollo de MIDAS para el aspecto de comportamiento.

4.6 Conclusiones

En este capítulo se ha descrito MIDAS, un marco metodológico dirigido por modelos para el desarrollo ágil de SIW. MIDAS propone una *arquitectura dirigida por modelos* basada en la arquitectura MDA (*Model Driven Architecture*) propuesta por el OMG (*Object Management Group*) y un *proceso de desarrollo ágil* basado en el uso de técnicas procedentes de metodologías como XP (*eXtreme Programming*) y AM (*Agile Modeling*). De este modo, el marco propuesto permite la realización de desarrollos ágiles dirigidos por modelos, facilitando de este modo la creación de software de forma iterativa e incremental gracias a la generación de código simultáneamente con los modelos.

MIDAS propone modelar el SIW de acuerdo con los aspectos de contenido, hipertexto y comportamiento en los niveles de CIM, PIM y PSM. Este trabajo se ha centrado, por un lado, en el aspecto de hipertexto, presentando el proceso de desarrollo a nivel de PIM; y, por otro lado, en el aspecto de contenido presentando el proceso de desarrollo de una BD web a nivel de PIM y PSM.

Para el desarrollo del hipertexto MIDAS, a diferencia de las metodologías tradicionales que proponen el modelado del hipertexto desde un punto de vista estructural, se ha presentado una propuesta orientada a servicios de usuarios; es decir, basada principalmente en los servicios que el usuario requiere del sistema y partiendo de un modelo de casos de uso específico en el cual se representan dichos servicios. En este trabajo se ha presentado el método propuesto para el modelado del hipertexto, describiendo los modelos definidos a nivel de PIM y las reglas de transformación entre estos modelos.

Para el desarrollo del aspecto del contenido se han presentado los modelos a nivel de PIM y de PSM tanto para la tecnología OR como para XML, así como las reglas de transformación para pasar de un nivel de PIM al PSM correspondiente. Además se han incluido las dos extensiones a UML necesarias para poder modelar todo el sistema en una notación única.

Para validar la metodología propuesta por MIDAS para el desarrollo de SIW, se han desarrollado diferentes casos de estudio que han permitido el refinamiento sucesivo de los diferentes modelos, extensiones y reglas de transformación para la generación (semi-)automática de un SIW.

Además, se está implementando una herramienta CASE (MIDAS-CASE), que integra todas las técnicas propuestas en MIDAS para la generación (semi-)automática de un SIW. El repositorio de esta herramienta CASE está siendo implementado utilizando una BD XML, concretamente utilizando XML DB de Oracle [44] y siguiendo la aproximación propuesta en este trabajo.

4.7 Agradecimientos

Esta investigación se ha llevado a cabo en el marco del proyecto DAWIS, financiado por el Ministerio de Ciencia y Tecnología (TIC 2002-04050-C02-01) y del proyecto PPR-2004-15 financiado por la Universidad Rey Juan Carlos.

4.8 Referencias

1. ACR-NEMA. *The DICOM Standard*. Recuperado de: http://medical.nema.org/, 2003.
2. Acuña C., Marcos E., de Castro V. y Hernández J.A. *A Web Information System for Medical Images Management*. Fifth International Symposium of Biological and Medical Data Analysis (ISBMDA 2004). Barcelona, LNCS 3337, Springer-Verlag, pp. 49-59, noviembre 2004.
3. Acuña C., Marcos E., de Castro V. y Hernández J.A. *Managing Medical Images*. Ercim News, Vol. 58 pp. 54-55, julio 2004.
4. Ambler, S. *Persistence Modeling in the UML.* 1999. En: http://www.sdmagazine.com/documents/s=755/sdm9908q/9908q.htm.
5. Ambler, S. *Agile Model Driven Development is Good Enough.* IEEE Software, pp. 71-73, septiembre-octubre 2003.
6. Ambler, S.W., Jeffries, R. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process.* Ed.Wiley. 2002.

7. Atkinson, C. y Kühne, T. *Model-Driven Development: A Metamodeling Foundation*. IEEE Software, Vol. 4, pp. 36-41, septiembre-octubre de 2003.
8. Atzeni, P., Merialdo, P. y Mecca, G. *Data-Intensive Web Sites: Design and Maintenance*. World Wide Web 4(1-2), pp. 21-47, 2001.
9. Beck, K. *Extreme Programming Explained: Embrace Change.* Ed.Addison-Wesley 2001.
10. Brambilla, M. *Extending hypertext conceptual models with process-oriented primitives*, ER 2003 (International Conference on Conceptual Modeling), octubre 2003, Chicago.
11. Bonifati, A., Stefano, C., Fraternali, P. y Maurino, *A. Building Multi-device, Content-Centric Applications Usign WebML and the W3I3 Tool Suite. Conceptual Modeling for E-Bussiness and the Web*. Ed. S. W. Liddle, H. C. Mayr, B. Thalheim, Springer Verlag, pp. 64-75, 2000.
12. Cáceres, P., Marcos, E., de Castro, V. *Navigation Modeling From A User Service Oriented Approach.* Proceedings of the Third Biennial International Conference in Advanced in Information Systems (ADVIS 2004). Tatyana Yakhno Ed. LNCS 3271. Springer Verlag, pp. 150-160, 2004.
13. Cáceres, P., Marcos, E., Díaz, F. *Agile Model Driven Development in Web Information Systems: A Case Study.* Proceedings of Thirtheenth International Conference on Information Systems Development (ISD 2004). Eds. O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic y S. Wrycza, pp. 341-351,2004.
14. Cáceres, P., Marcos, E. y Vela, B. *A MDA-Based Approach for Web Information System Development*. Workshop in Software Model Engineering in UML Conference. San Francisco, USA, octubre, 2003.
15. Castano, S., Palopoli, L., Torlone, R. *A General Methodological Framework for the Development of Web-Based Information Systems*. Workshop in Conceptual Modeling for E_Business and the Web (ER 2000). Liddle, S. W., Mayr, H. C., Thalheim, B. (eds.). LNCS 1921. Springer Verlag, 2000.
16. Castro de V., Marcos, E., Cáceres, P. *A User Service Oriented Method to model Web Information Systems*. Porceeding of the International Conference on Web Information Systems Engineering (WISE 2004). Ed. Xiaofang Zhou, Stanley Su, Mike P. Papazoglou, Maria E. Orlowska, Keith G. Jeffery, Springer Verlag, pp. 41-52, 2004.
17. Castano, S., Palopoli, L. y Torlone, R. *A General Methodological Framework for the Development of Web-Based Information Systems. Conceptual Modelling for E_Bussines and the Web*. Ed. S. W. Liddle, H. C. Mayr y B. Thalheim. Springer-Verlag, Lecture Notes in Computer Science, Berlin, 2000.
18. Conallen, J. *Building Web Applications with UML*. Addison Wesley, 2000.
19. ConfMaster. Disponible en http://confmaster.net/phpwebsite_en/index.php, 2004.
20. Díaz, P., Aedo, I. y Montero, S. *Ariadne, a Development Method for Hypermedia*. DEXA 2001, pp. 764-774, 2001.
21. Dori, D. *Conceptual Modeling and System Architecting.* Communications of the ACM, Vol. 46, nº 10, pp. 63-65, octubre 2003.
22. Eisenberg, A., Melton, J. Kulkarni, K., Michels, J. y Zemke F. *SQL:2003 has been published* ACM SIGMOD Record, Vol. 33, No. 1, pp. 119-126, 2004.
23. Fraternali, P. y Paolini, P. *A conceptual Model and a Tool Environment for Developping more Scalable, Dynamic and Customizable Web Applications.* Advances in Database Technology. Ed. Sheck, Saltor, Ramos, Alonso. Proceedings of the 6th. Conference on Extended Database Technology (EDBT'98). Springer-Verlag. Valencia, España, 1998.
24. Fraternali, P.. *Tools and approaches for developing data-intensive Web applications: A survey.* ACM Computing Surveys, Volumen 31, nº 3, 1999.
25. Fowler, M. *The New Methodology.* Recuperado en mayo 2001 de: http://www.martinfowler.com/articles/newMethodology.html.
26. Garzotto, F., Paolini, P. y Schwabe, D. (1993). HDM - a Model-Based Approach to Hypertext Application Design. ACM TODS, enero 1993, 11(1), pp. 1-26, 1993.

27. Gómez, J., Cachero, C., Pastor, O. *Conceptual Modeling of Device-Independent Web Applications*. IEEE Multimedia, 8 (2), pp. 26-39, 2001.
28. Hennicker, R., Koch, N. *A UML-based Methodology for Hypermedia Design*. UML' 2000, LNCS 1939, pp.410-424, 2000.
29. Insfrán, E., Pelechano, V. y Pastor, O. *Conceptual Modeling in the eXtreme.* Information & Software Technology, 44(11), pp. 659-669, 2002.
30. Isakowitz, T., Kamis, A., Koufaris, M.: The Extended RMM Methodology for Web Publishing. Working Paper IS-98-18, Center for Research on Information System. Retrieved from: http://rmm-java.stern.nyu.edu/rmm/, 1998.
31. Koch, N., Kraus, A., Cachero, C., Meliá, S. *Modeling Web Business Processes with OO-H and UWE*. International Workshop on Web-oriented Software Technology (IWWOST). Schwabe, D., Pastor, O., Rossi, G., Olsina, L. (eds.), 27-50, julio 2003.
32. Kulkarni, V. y Reddy, S. *Separation of Concerns in Model-Driven Development*. IEEE Software, Vol. 4, pp. 64-69, septiembre-octubre, 2003.
33. Marcos, E., Cáceres, P, y De Castro, V. *From the Use Case Model to the Navigation Model: a Service Oriented Approach*. CAISE FORUM '04. Riga (Letonia). 10-11 junio 2004. Ed: J. Grabis, A. Persson y J. Stirna. Proceedings, 2004.
34. Marcos, E., De Castro, V. y Vela, B. *Representing Web Services with UML: A Case Study*. The First International Conference on Service Oriented Computing (ICSOC03). Trento (Italy), LNCS 2910, Springer Verlag, pp. 17-27, diciembre, 2003.
35. Marcos E., Vela B. y Cavero, J. M. *Extending UML for Object-Relational Database Design*. Fourth Int. Conference on the Unified Modeling Language, UML 2001. Toronto (Canadá), LNCS 2185, Springer-Verlag, pp. 225-239, octubre, 2001.
36. Marcos, E. Vela, B., Cáceres, P. y Cavero, J.M. *MIDAS/DB: a Methodological Framework for Web Database Design*. DASWIS 2001. Yokohama (Japón), noviembre, 2001. LNCS 2465, Springer-Verlag, pp. 227-238, septiembre, 2002.
37. Marcos, E., Vela, B. y Cavero J.M. *Methodological Approach for Object-Relational Database Design using UML*. Journal on Software and Systems Modeling (SoSyM). Springer-Verlag. Ed.: R. France y B. Rumpe. Vol. SoSyM 2, pp.59-72, 2003.
38. Mayo Clinic, *Analyze Software*, Recuperado de: http://www.mayo.edu/ bir/Software/ Analyze/ Analyze.html, 2004.
39. Mecca, G., Merialdo, P., Atzeni, P. y Crescenzi, V (1999). The ARANEUS guide to Web-site development.. En: http://poincare.inf.uniroma3.it/
40. OMG. *UML Superstructure 2.0. Draft adopted Specification*. Recuperado de: http://www.uml.org/.
41. OMG. *Unified Modeling Language Specification*. Version 1.5. Recuperado de: http://www.omg.org/technology/documents/formal/uml.htm, 2003.
42. OMG. *Model Driven Architecture*. Document number ormsc/2001-07-01. Ed.: Miller, J. y Mukerji, J. Recuperado de: http://www.omg.com/mda, 2001.
43. OMG. *MDA Guide Version 1.0*. Document number omg/2003-05-01. Ed.: Miller, J. y Mukerji, J. Recuperado de: http://www.omg.com/mda, 2003.
44. Oracle Corporation. *Oracle XML DB. Technical White Paper*. Recuperado de: www.otn.com, enero, 2003.
45. Papazoglou, M.P. y Georgakopoulos, D. *Serviced-Oriented Computing*. Communications of ACM, Volume: 46, 10, octubre, 2003, pp. 25-28.
46. Retschitzegger, W. y Schwinger W. *Towards Modeling of Data Web Applications- A Requirement's Perspective.* En Proceedings of the America's Conference on Information Systems. V.I, pp. 149-155, 2000.
47. Schwabe, D. y Rossi, G. (1995). The Object-Oriented Hypermedia Design Model. Communications ACM, agosto 1995, 58(8), pp. 45-46, 1995.
48. Schwabe, D. y Rossi, G.(1998). *An object oriented approach to web-based applications design*. Theory and practice of object system, 4 (4), pp. 207-225, 1998.

49. Turk, D., France, R., Rumpe, B. y Georg, G. *Model driven Approach to Software Development.* Advances in Object-Oriented Information Systems. LNCS 2426, pp.229-230, septiembre, 2002.
50. Vara, J.M., de Castro, V., Cáceres P. y Marcos E. *Arquitectura de MIDAS-CASE: una herramienta para el desarrollo de SIW basada en MDA*. JIISIC'04 en Actas de las IV Jornadas Iberoamericanas en Ingeniería del Software e Ingeniería del Conocimiento. Ed. O. Dieste y A. M. Moreno. Madrid, 3-5 de noviembre de 2004.2004, pp.83-92, 2004.
51. Vela, B. y Marcos E. *Extending UML to represent XML Schemas*. The 15th Conference On Advanced Information Systems Engineering. CAISE'03 FORUM. Klagenfurt/Velden (Austria). 16-20 junio 2003. Ed: J. Eder y T. Welzer. Short Paper Proceedings, 2003.
52. Vela, B., Acuña, C. y Marcos, E. *A Model Driven Approach for XML Database Development*, 23rd. International Conference on Conceptual Modelling (ER2004). LNCS 3288. Springer Verlag, pp. 780-794. 2004.
53. Weneger, H. *Agility in Model Driven Software Development? Implications for Organization, Process and Architecture.* 2002. Recuperado de: http://www.softmetaware.com/oopsla2002/wenegerh.pdf.
54. W3C. *Web Services Description Language (WSDL)* Version 2.0 Part 1: Core Language. W3C Working Draft 3 agosto 2004. Recuperado de: http://www.w3.org/TR/2004/WD-wsdl20-20040803/, 2004
55. W3C. XML *Extensible Markup Language (XML) 1.0 (Third Edition).* W3C Recommendation. Bray, T., Paoli, J, Sperberg-McQu4een, C. M., Maler, E. y Yergeau F. Recuperado de: http://www.w3.org/TR/2004/REC-xml-20040204/, 2004.
56. W3C XML Schema Working Group. *XML Schema Parts 0-2:[Primer, Structures, Datatypes]*. W3C Recommendation. Recuperado de: http://www.w3.org/TR/xmlschema-0/, http://www.w3.org/TR/xmlschema-1/ y http://www.w3.org/TR/xmlschema-2/, 2001.

5 El Desarrollo Documental

José Luis Sierra, Alfredo Fernández-Valmayor, Baltasar Fernández-Manjón, Antonio Navarro

Dpto. Sistemas Informáticos y Programación. Fac. Informática. Universidad Complutense de Madrid.
C/ Profesor José García Santesmases S/N. 28040 Madrid
{jlsierra,alfredo,balta,anavarro}@sip.ucm.es

5.1 Introducción

Una de las principales características de una gran parte de las aplicaciones web es el integrar vastas colecciones de contenidos referidos a un determinado dominio o área de conocimiento, y que son proporcionadas y mantenidas por expertos en dicho dominio. Este tipo de aplicaciones son un caso particular de las que en este capítulo se denominarán **aplicaciones ricas en contenidos**, aplicaciones orientadas al procesamiento y a la presentación interactiva de contenidos, cuyo principal factor de variabilidad está determinado por la diversidad de los contenidos integrados en las mismas.

El éxito en el desarrollo de una aplicación rica en contenidos depende en gran medida de la eficiencia de los mecanismos que regulan la comunicación entre los diferentes actores involucrados en dicho desarrollo: los **expertos en el dominio**, que son los propietarios y/o los autores de los contenidos, y los **desarrolladores de software**, expertos informáticos responsables de la construcción de la aplicación final. Las tecnologías de marcado descriptivo proporcionan mecanismos eficientes para orquestar dicha comunicación. Efectivamente, estas tecnologías permiten articular la comunicación mediante el intercambio de documentos que pueden ser entendidos tanto por los expertos como por los desarrolladores, y que pueden ser automáticamente procesados por estos últimos durante la producción y el mantenimiento de las aplicaciones. De esta forma, los expertos en el dominio proporcionan a los desarrolladores documentos con los contenidos de la aplicación. Por su parte, los desarrolladores documentan otras características operacionales de la aplicación no directamente derivables de los contenidos (v.g. propiedades configurables de su interfaz de usuario), lo que facilita a los expertos en el dominio la comprensión y posterior mantenimiento de dichas características. Todos estos documentos se marcan mediante un **lenguaje de marcado específico del dominio** (LMED), especialmente diseñado para representar el modelo de aplicación que se quiere construir y para facilitar a los expertos en el dominio su comprensión y su uso. Dicho marcado posibilita la producción de la aplicación final mediante el procesamiento de dichos documentos con un procesador apropiado para dicho LMED. Este protocolo constituye en esencia el **desarrollo documental** de aplicaciones ricas en contenidos descrito en este capítulo.

Los autores de este capítulo han aplicado satisfactoriamente el desarrollo documental en la construcción de varias aplicaciones educativas e hipermedia, así como en la construcción de sistemas basados en el conocimiento. Como fruto de estas experiencias se ha formulado la **Aproximación Documental al Desarrollo de Software** (ADDS), un modelo de proceso que regula y sistematiza este tipo de desarrollo. Así mismo, dichas experiencias han puesto de manifiesto que los documentos con los contenidos y las características operacionales y, por tanto, las necesidades de marcado de los expertos y los desarrolladores, normalmente varían durante el proceso de desarrollo. Por tanto es fundamental llevar a cabo una formulación incremental de los LMEDs a fin de acomodar las nuevas exigencias de marcado conforme éstas aparecen. Para tal fin, y en el contexto de ADDS, se ha desarrollado la técnica de **Provisión de LMEDs en ADDS** (PADDS). Así mismo, la formulación incremental de LMEDs conduce a una construcción incremental de sus procesadores. Dicha construcción se regula en ADDS mediante el modelo de **Operacionalización en ADDS** (OADDS), un modelo conceptual para la construcción incremental de procesadores para LMEDs en ADDS.

El capítulo comienza analizando la racionalidad y los principios básicos del desarrollo documental de aplicaciones ricas en contenidos (sección 5.2). Seguidamente se presenta ADDS, el modelo de proceso que sistematiza y regula dicho desarrollo (sección 5.3). A continuación se describen PADDS, la técnica para la formulación incremental de LMEDs en ADDS (sección 5.4) y OADDS, el modelo incremental de operacionalización seguido en ADDS (sección 5.5). Para finalizar, se analizan diferentes trabajos relacionados con el aquí expuesto (sección 5.6), y se proporcionan las principales conclusiones obtenidas así como algunas líneas de trabajo futuro (sección 5.7).

5.2 Visión general del desarrollo documental

La metodología de desarrollo documental presentada en este capítulo se sugirió inicialmente en [6,7] como una estrategia para facilitar la producción y mantenimiento de las aplicaciones educativas, y se consolidó a lo largo de varias experiencias en el desarrollo de aplicaciones ricas en contenidos en varios dominios. En [8] esta metodología se aplica en la producción y mantenimiento de aplicaciones educativas para la comprensión de textos escritos en lenguas próximas similares a la del estudiante (más concretamente, en el contexto de las lenguas romances). En [25,26] la metodología se aplica para dar soporte a la generación de prototipos para aplicaciones hipermedia, mientras que en [31] se propone su uso para la construcción de sistemas basados en el conocimiento. En esta sección se motiva el desarrollo documental y se proporciona una visión general de la metodología. De esta forma, se comienza caracterizando las principales dificultades que surgen durante la producción y mantenimiento de las aplicaciones ricas en contenidos, y se muestra cómo dichas dificultades son principalmente debidas a la falta de eficiencia en la comunicación entre expertos en el dominio y desarrolladores (subsección 5.2.1). Seguidamente se muestra cómo en el dominio más restringido de la edición de documentos electrónicos surgen dificultades análogas, y cómo dichas dificultades pueden ser

satisfactoriamente subsanadas mediante el uso de tecnologías de marcado descriptivo (subsección 5.2.2). A continuación se muestra cómo dichas soluciones se pueden aplicar en la producción y el mantenimiento de las aplicaciones ricas en contenidos, dando lugar al desarrollo documental de dichas aplicaciones (subsección 5.2.3). Para finalizar, la subsección 5.2.4 esboza un ejemplo de desarrollo documental.

5.2.1 La problemática de la producción y el mantenimiento de las aplicaciones ricas en contenidos

La producción y el mantenimiento de aplicaciones ricas en contenidos son tareas costosas. Aunque existen aplicaciones que pueden ser directamente producidas y mantenidas por los expertos en el dominio utilizando herramientas de autor apropiadas, conforme crece la complejidad de los contenidos y la sofisticación del procesamiento de dichos contenidos dicho enfoque se vuelve inviable. En este caso es necesario involucrar explícitamente un equipo de desarrolladores informáticos que haga frente a las complejidades técnicas de la producción y del mantenimiento de dichas aplicaciones. El proceso típico de desarrollo de este tipo de aplicaciones procede a través de varias iteraciones, en las que los expertos proporcionan los contenidos y los desarrolladores producen nuevas versiones de la aplicación. Dichas versiones son evaluadas por los expertos, que proponen modificaciones y/o mejoras a los desarrolladores [9,16]. Este escenario se esquematiza en la Figura 5.1.

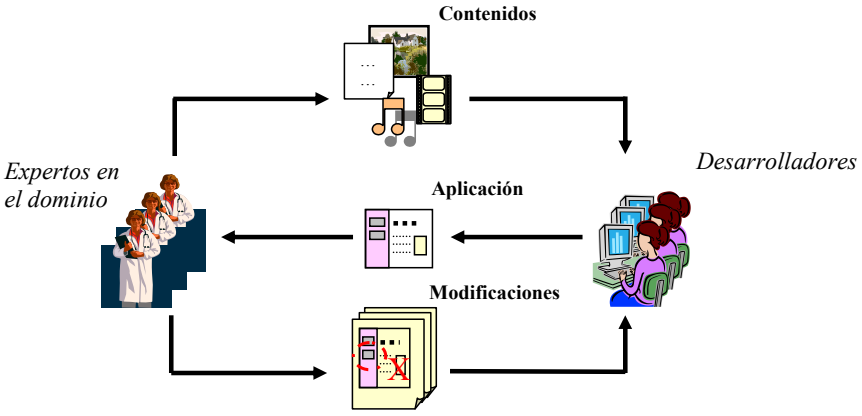


Figura 5.1 Colaboración entre los expertos en el dominio y los desarrolladores durante la producción y mantenimiento de una aplicación rica en contenidos.

La principal ventaja de escenarios de desarrollo como el mostrado en la Figura 5.1, donde se involucra explícitamente a comunidades de expertos y desarrolladores, es su flexibilidad, ya que se evitan las limitaciones expresivas impuestas por la utilización de herramientas de autor concretas. Dichas herramientas son sustituidas por informáticos capaces de responder y adaptarse a las necesidades de los expertos en el dominio conforme éstas surgen. Tal y como los autores de este capítulo han comprobado en experiencias como las descritas en [6,7,8], dicha flexibilidad

comienza a ser crítica en el desarrollo de aplicaciones de tamaño medio. Sin embargo, dichos escenarios deben vencer varias dificultades:

- La **dificultad de comunicación entre expertos y desarrolladores**. Es necesario articular mecanismos apropiados que permitan a los expertos proporcionar los contenidos a los desarrolladores, así como proponer a estos las modificaciones y mejoras requeridas en las aplicaciones. En ausencia de dichos mecanismos, los desarrolladores se verán obligados a incorporar manualmente en la aplicación los contenidos proporcionados por los expertos. Por su parte los expertos en el dominio deberán comunicar informalmente sus modificaciones y mejoras a los desarrolladores (v.g. directamente sobre capturas impresas de las pantallas de la aplicación [8]).
- La **dificultad de reutilización de los contenidos** y la **brevedad del ciclo de vida**. Los contenidos proporcionados por los expertos pueden estar representados en formatos heterogéneos y/o propietarios, siendo incluso habitual el uso de contenidos en formato impreso. Así mismo, dichos contenidos pueden ser de nuevo codificados por los desarrolladores en formatos dependientes de la aplicación y orientados a facilitar el proceso de los mismos. Esta heterogeneidad dificulta su reutilización en nuevas aplicaciones o, incluso, en versiones futuras de una misma aplicación. El resultado es un defecto muy habitual de las aplicaciones ricas en contenidos: la brevedad de su ciclo de vida. Efectivamente, cuando las tecnologías evolucionan, las aplicaciones se quedan obsoletas, se abandonan, y el valor de los contenidos integrados en las mismas se diluye como consecuencia de la dificultad de su reutilización.
- La **dificultad de reutilización de los programas**. En muchas situaciones los programas que procesan y presentan los contenidos son muy dependientes de la aplicación particular. Esto dificulta su reutilización en el desarrollo de otras aplicaciones.

La superación de estas dificultades puede llevarse a cabo haciendo explícita la relación existente entre la estructura de los contenidos de la aplicación y otras características potencialmente configurables de la misma mediante una representación que cumpla los siguientes requisitos:

- **Requisito de comprensibilidad**. La representación debe ser comprensible por los expertos en el dominio, a fin de que estos puedan llevar a cabo directamente la producción y el mantenimiento de los aspectos expuestos.
- **Requisito de procesabilidad**. La representación debe ser procesable, permitiendo a los desarrolladores la automatización de su traducción a formatos más convenientes para el procesamiento y la presentación en el seno de la aplicación.
- **Requisito de reusabilidad**. La representación debe facilitar la reutilización de los aspectos expuestos, y fundamentalmente de los contenidos, tanto en futuras versiones de la misma aplicación como en la construcción de nuevas aplicaciones.

La elección de una representación comprensible y procesable sienta las bases para una comunicación efectiva entre expertos en el dominio y desarrolladores. Los primeros pueden proporcionar directamente los contenidos y las modificaciones en dicha representación. Los últimos pueden reconstruir automáticamente la aplicación a partir de la información proporcionada. Este último hecho también mejora la

reutilización de los programas, ya que dichos programas se estructuran ahora como una *shell* que permite generar los distintos miembros de una misma familia de aplicaciones una vez fijados adecuadamente sus parámetros de variabilidad. Por último, la elección de una forma de representación reutilizable permite alargar el ciclo de vida de las aplicaciones y reutilizar sus contenidos. El desarrollo documental emplea lenguajes de marcado descriptivo como base de la técnica de representación escogida. Esta elección encuentra su motivación en las técnicas utilizadas en la publicación moderna de documentos electrónicos que se analizan brevemente en la próxima subsección.

5.2.2 El dominio de la publicación de documentos electrónicos

La publicación de documentos electrónicos presenta marcadas similitudes con la producción y el mantenimiento de aplicaciones ricas en contenidos. La Figura 5.2a muestra un escenario de publicación muy simplificado que pone de manifiesto dichas similitudes. De acuerdo con dicho escenario, el proceso de publicación implica dos tipos de participantes con conocimientos y habilidades muy diferentes: los **`autores`** de los originales que deben ser publicados, y los **`editores`** encargados de publicar dichos originales. En la Figura 5.2a se muestra un autor que proporciona su original al editor, quién produce una primera impresión de éste. El autor entonces examina dicha impresión y, eventualmente, señala en la misma una serie de correcciones. En base a estas correcciones el editor produce nuevas impresiones corregidas, lo que conduce a una situación análoga a la que se esquematiza en la Figura 5.1. No obstante, en los entornos de publicación electrónica modernos existen procedimientos bien establecidos que permiten acelerar este proceso de producción y corrección de pruebas.

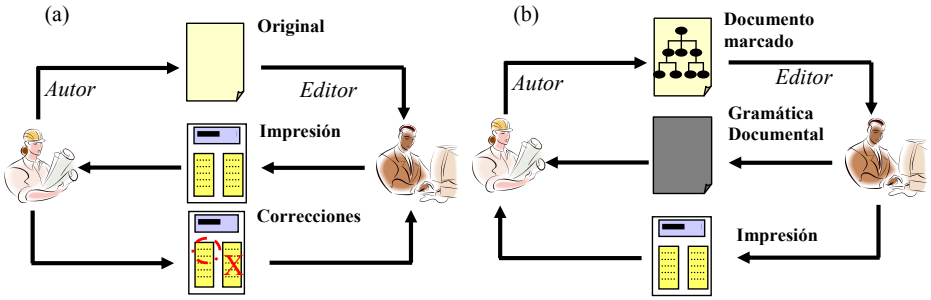


Figura 5.2 (a) Colaboración entre un autor y un editor durante la publicación de un original; (b) mejora del proceso mediante el uso de lenguajes de marcado descriptivo.

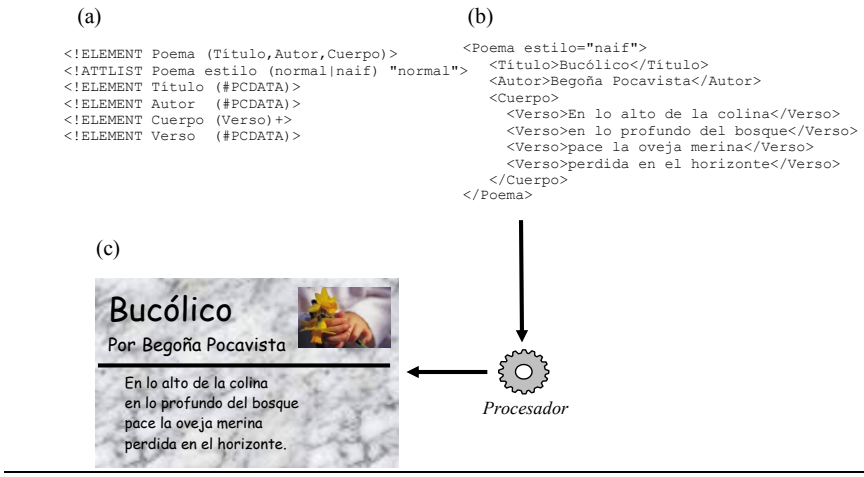


Figura 5.3 (a) Una DTD muy simple para el marcado de poemas; (b) un documento válido respecto a la DTD en (a); (c) el documento en (b) puede procesarse con un procesador apropiado para obtener impresiones.

El uso de lenguajes de marcado descriptivo [3,11,12] constituye un procedimiento ampliamente aplicado en el dominio de la publicación de documentos electrónicos para mejorar el bucle de comunicación entre autores y editores. La Figura 5.2b resume el proceso así mejorado. En este escenario el editor proporciona al autor un lenguaje de marcado descriptivo definido mediante una **`gramática documental`** (v.g. una DTD, *Document Type Definition*, como la que se muestra en la Figura 5.3a). La naturaleza descriptiva de este lenguaje implica que el marcado impuesto por el mismo está orientado a representar la estructura lógica de los documentos en lugar de cómo dichos documentos deben ser posteriormente procesados (v.g. véase la Figura 5.3b). De esta forma, el autor es capaz de comprender este lenguaje y de utilizar el mismo para explicitar la estructura de su original mediante marcado descriptivo. Así mismo, el autor puede comprobar la corrección del marcado añadido validando el

original marcado con la gramática documental utilizada (v.g. el documento de la Figura 5.3b puede validarse con la DTD de la Figura 5.3a). Por su parte, el editor tomará el original validado y lo procesará para producir las impresiones. Dado que el documento se ajusta al lenguaje de marcado, los posibles defectos que prevalecen en dichas impresiones son debidos, bien a errores en los contenidos, bien a un uso intencionalmente incorrecto del marcado, aunque formalmente correcto respecto a la gramática documental que define el lenguaje. En cualquier caso, ambos tipos de defectos pueden ser directamente subsanados por el autor sobre el original. De esta forma, el editor puede producir nuevas impresiones sin coste apreciable. De hecho, las impresiones pueden ser automáticamente generadas a partir del original marcado procesando el mismo con un **`procesador`** adecuado para el lenguaje de marcado utilizado (Figura 5.3c).

Es interesante observar cómo las representaciones de documentos basadas en lenguajes de marcado descriptivo satisfacen los criterios de comprensibilidad, procesabilidad y reutilización cuando éstas se aplican al dominio de la publicación de documentos electrónicos [3]. De hecho, los lenguajes de marcado descriptivo pueden ser fácilmente entendidos por los autores, que pueden utilizarlos directamente, o bien con la ayuda de editores especializados de tipo "lo que ves es lo que tienes" (WYSIWYG, *What You See Is What You Get*). Así mismo, dado que dichos lenguajes pueden formalizarse mediante gramáticas documentales, el procesamiento de los originales marcados puede automatizarse. Por último, un mismo documento marcado descriptivamente puede utilizarse con diferentes propósitos y sobre diferentes dispositivos (por ejemplo, para generar una presentación web o para generar un fichero *postscript* orientado a obtener una copia impresa del documento). Para ello basta con cambiar el procesador utilizado para atribuir significado de presentación al marcado descriptivo. De esta forma, es natural plantear la generalización de esta representación para el desarrollo de aplicaciones ricas en contenidos, dando lugar al desarrollo documental de este tipo de aplicaciones. Esta generalización se detalla en la siguiente subsección.

5.2.3 Desarrollo documental de aplicaciones ricas en contenidos

El desarrollo documental de una aplicación rica en contenidos propugna la formulación de una vista documental de sus contenidos y del resto de sus características configurables, es decir, de sus características operacionales. Por tanto, todos estos aspectos pueden describirse mediante un conjunto de documentos que se marcan con un LMED descriptivo y específico para este tipo de aplicación. Estos documentos pueden ser comprendidos, producidos y mantenidos por los expertos en el dominio, ya que son específicos de dicho dominio de aplicación. Así mismo, dichos expertos pueden entender y aplicar el LMED utilizado en su marcado, puesto que dicho LMED se diseña explícitamente para representar la estructura lógica de dichos documentos y el modelo de aplicación que se pretende construir. Así mismo, el marcado descriptivo añadido a los documentos posibilita su procesamiento automático a fin de reconstruir la aplicación cuando dichos documentos cambian. De esta manera, este enfoque proporciona un balance razonable entre los requisitos de comprensibilidad y de procesabilidad. Igualmente, el enfoque también propugna la

reutilización de los contenidos, dado que en la formulación del LMED pueden aplicarse estándares bien establecidos, como el **`lenguaje de marcado generalizado estándar`** (SGML, *Standard Generalized Markup Language*) [11] o el **`lenguaje de marcado extensible`** (XML, *eXtensible Markup Language*) [38].

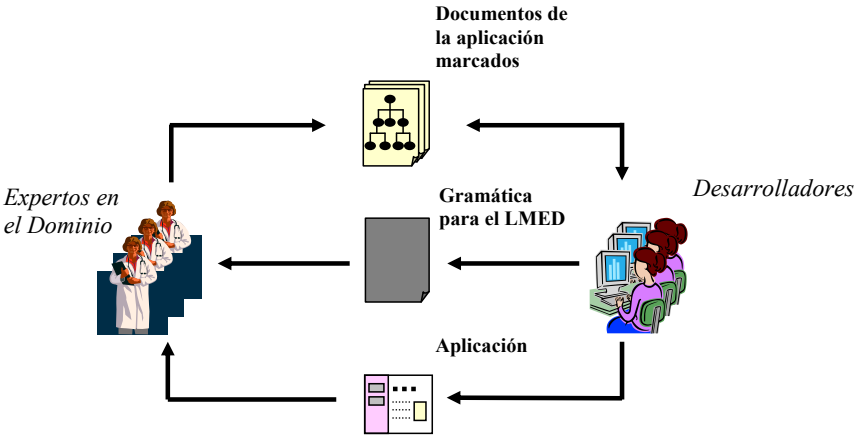


Figura 5.4 Desarrollo documental de aplicaciones ricas en contenidos: una primera aproximación.

La Figura 5.4 esquematiza una primera aproximación a un escenario de desarrollo documental. Dicho escenario es análogo al del uso de lenguajes de marcado descriptivo en el dominio de la publicación de documentos electrónicos mostrado en la Figura 5.2b. Es importante notar que los documentos utilizados en esta aproximación no están restringidos a prosa textual, sino que incluyen los contenidos de la aplicación y las características operacionales de la misma. Tal y como se ha indicado anteriormente, estos **`documentos de la aplicación`** se marcan con un **`LMED de la aplicación`** que es específico para cada familia de aplicaciones. Las aplicaciones en sí se producen automáticamente como resultado de procesar dichos documentos marcados con procesadores apropiados, de la misma forma que las impresiones se producen a partir de los originales marcados en la Figura 5.2b. Obsérvese que los desarrolladores también juegan un papel activo en la autoría de los documentos de la aplicación. De hecho, dichos desarrolladores pueden documentar un cuerpo inicial mínimo de contenidos, así como el resto de características operacionales. Esto permite la producción de un prototipo inicial de la aplicación, así como la generación de unas plantillas sobre las que los expertos en el dominio pueden trabajar inicialmente. Así mismo, los desarrolladores también pueden ayudar a los expertos en el uso del LMED para el marcado de los documentos.

La filosofía de desarrollo documental satisface los requisitos de comprensibilidad, procesabilidad y reutilización enunciados en la subsección 5.2.1. Sin embargo, dicha filosofía adolece de un coste inicial de aplicación elevado, debido a que se basa en una formulación explícita de un lenguaje, el LMED de la aplicación, así como en la construcción explícita de un procesador para dicho lenguaje. La adopción de una estrategia incremental de definición y operacionalización de lenguajes permite

amortizar este coste a lo largo del proceso de desarrollo de toda una familia de aplicaciones similares. De esta forma, en lugar de concebir los LMEDs como entidades inamovibles, prefijadas a priori, dichos lenguajes deben considerarse como objetos dinámicos que evolucionan durante el proceso de desarrollo de aplicaciones en un determinado dominio, adaptándose a las necesidades de marcado de los expertos y los desarrolladores conforme dichas necesidades se ponen de manifiesto. Esta evolución no afecta únicamente a la naturaleza descriptiva de los LMEDs, sino también a sus aspectos operacionales, es decir, a sus procesadores. Por tanto, el éxito del desarrollo documental depende en gran medida de una gestión adecuada de la definición incremental de los LMEDs y del desarrollo incremental de sus procesadores. Todas estas consideraciones se reflejan en el modelo de proceso ADDS, que se detalla en la sección 5.3, así como en la técnica PADDS para la provisión incremental de LMEDs (sección 5.4) y en el modelo de operacionalización OADDS (sección 5.5). Antes de abordar tales aspectos se presenta un ejemplo para plasmar de forma práctica las principales ideas del desarrollo documental.

5.2.4 Un ejemplo de desarrollo documental

En esta subsección se esboza brevemente un ejemplo de desarrollo documental de un tipo de aplicación orientada a la recomendación de rutas en redes de metro. En este ejemplo, los expertos en el dominio son expertos en la gestión de redes de transporte. Por su parte, los contenidos pueden concebirse como un conjunto de tablas que proporcionan una representación de los aspectos estructurales y dinámicos (v.g. velocidades medias en líneas, tiempos de trasbordo medios en estaciones, etc.) de dichas redes. Durante el mantenimiento de este tipo de aplicaciones puede modificarse la representación de la información de la red, y también otras características operacionales configurables en la aplicación (v.g. los mensajes informativos en la interfaz de usuario).

De esta forma, las características variables de una aplicación para la recomendación de rutas pueden representarse mediante un documento que contiene la descripción de la red, tanto en lo que se refiere a los aspectos estructurales, como en lo referido a sus aspectos dinámicos, así como características operacionales relativas a atributos configurables de su interfaz de usuario (v.g. los nombres de los botones, el texto de las etiquetas, mensajes informativos, una referencia a la imagen con el mapa de la red, y las coordenadas de las estaciones en dicha imagen). En base a estas consideraciones, los desarrolladores pueden proporcionar un LMED para marcar todos estos aspectos. En la Figura 5.5a se muestra un fragmento de DTD para este LMED. Nótese que el vocabulario de marcado y la estructura de este LMED se eligen para facilitar su comprensión por parte de los expertos en gestión de redes. De esta forma estos expertos pueden producir, modificar y mantener el documento de la aplicación, contando con la ayuda de los desarrolladores informáticos si ello fuera necesario.

(a)

```
<!ELEMENT Metro      (Red,InterfazDeUsuario)>
<!ELEMENT Red        (Estructura,Dinámica)>
<!ELEMENT Estructura (Estaciones,Líneas)>
<!ELEMENT Estaciones (Estación)+>
<!ELEMENT Estación   (#PCDATA)>
<!ATTLIST Estación id ID #REQUIRED>
...
<!ELEMENT InterfazDeUsuario (Título,
                             BotónExit,
                             BotónReset,...,
                             Mapa,
                             Coordenadas)>
<!ELEMENT Título (#PCDATA)>
<!ELEMENT BotónExit (#PCDATA)>
...
<!ELEMENT Mapa EMPTY>
<!ATTLIST Mapa loc CDATA REQUIRED>
<!ELEMENT Coordenadas (Coordenada)+>
<!ATTLIST Coordenada estación IDREF #REQUIRED
                     x NMTOKEN #REQUIRED
                     y NMTOKEN #REQUIRED>
```

(b)

```
<Metro>
 <Red>
  <Estructura>
   <Estaciones>
    <Estación id="CONGOSTO">Congosto</Estación>
    <Estación id="VVALLECAS">Villa de Vallecas</Estación>
    ...
   </Estaciones>
   <Líneas>
   ...
   </Líneas>
  </Estructura>
  <Dinámica>
     ...
  </Dinámica>
 </Red>
 <InterfazDeUsuario>
   <Título>Búsqueda de caminos en redes de metro</Título>
   <BotónExit>Salir</BotónExit>
   <BotónReset>Volver a inicio</BotónReset>
   ...
   <Mapa loc="madrid.jpg"/>
   ...
   <Coordenadas>
     <Coordenada estación="CONGOSTO" x="690" y="406"/>
   ...
   </Coordenadas>
 </InterfazDeUsuario>
</Metro>
```

(c)

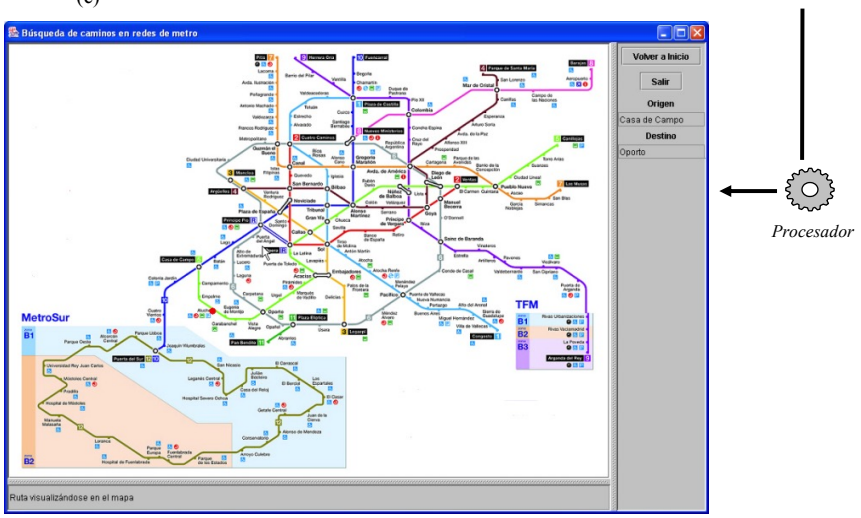


Figura 5.5 (a) Fragmento de la DTD para el LMED de las aplicaciones de recomendación de rutas en redes de metro; (b) fragmento del documento de la aplicación para la red de metro de Madrid; (c) aplicación para la red de metro de Madrid producida como resultado de procesar el documento en (b) con un procesador adecuado.

La Figura 5.5b muestra un fragmento del documento de la aplicación para la red de metro de Madrid. Este documento puede procesarse automáticamente para producir la correspondiente aplicación de recomendación de rutas (Figura 5.5c).

5.3 El modelo de proceso ADDS

En esta sección se detalla la Aproximación Documental al Desarrollo de Software (ADDS), un modelo de proceso para el desarrollo documental de aplicaciones ricas en contenidos. ADDS puede entenderse como una sistematización de dicho desarrollo. Por tanto, su principal objetivo es regular eficientemente la colaboración entre expertos y desarrolladores en la producción y mantenimiento de aplicaciones. Así mismo, ADDS presta especial atención a la viabilidad práctica del desarrollo documental. De esta forma propugna una estrategia iterativa e incremental para la producción y mantenimiento de las aplicaciones, de los LMEDs, y de los procesadores asociados con los mismos. El modelo de proceso ADDS se presenta desde tres perspectivas diferentes:

- **`Perspectiva de actividades y productos`**. Esta perspectiva analiza las principales actividades contempladas por el modelo, así como los productos producidos y consumidos por las mismas. Se detalla en la subsección 5.3.1.
- **`Perspectiva de secuenciación de actividades`**. Esta perspectiva analiza el modelo desde un punto de vista dinámico, haciendo énfasis en su carácter iterativo e incremental. Se desarrolla en la subsección 5.3.2.
- **`Perspectiva de actores y roles`**. Esta perspectiva (subsección 5.3.3) detalla los actores involucrados en el desarrollo, así como los roles que dichos actores juegan en las distintas actividades del modelo.

Estas perspectivas se ejemplifican en la subsección 5.3.4. Nótese que, siendo un modelo de proceso, ADDS no se compromete explícitamente con ningún procedimiento concreto para llevar a cabo las actividades contempladas, ni tampoco con tecnologías concretas para representar los productos involucrados. PADDS (sección 5.4) y OADDS (sección 5.5) son soluciones desarrolladas y probadas por los autores de este artículo que refinan alguno de los aspectos del modelo: la provisión de LMEDs y la construcción de sus procesadores. No obstante, el modelo en sí está abierto a cualquier otro tipo de soluciones, tal y como muestran las subsecciones que siguen.

5.3.1 La perspectiva de actividades y productos

La Figura 5.6 esquematiza las actividades y productos involucrados en el desarrollo documental de acuerdo con ADDS. En esta subsección se detallan cada uno de los aspectos reflejados por dicha perspectiva.

El objetivo de la actividad de **`Provisión del LMED`** es la obtención del LMED de la aplicación, que se define declarativamente mediante una gramática documental expresada en un formalismo apropiado (v.g. una DTD o bien cualquier otro lenguaje de esquema documental [21]). Dicha definición puede verse facilitada por la reutilización de LMEDs previamente definidos y almacenados en un **`repositorio de LMEDs`**. Durante esta actividad es necesario prestar especial atención al requisito de comprensibilidad. De esta forma, el vocabulario de marcas se debe elegir de modo que sea cercano al dominio de aplicación. Así mismo, la estructura impuesta por el marcado debe reflejar la estructura de la documentación

real manejada por los expertos en el dominio. Igualmente, la adhesión a un metalenguaje de marcado común (v.g. los ya citados SGML o XML) puede contribuir a mejorar dicha comprensibilidad, dado que todos los LMEDs compartirán una sintaxis común y homogénea, lo que evitará a los expertos en el dominio el tener que asimilar una nueva sintaxis con cada nuevo LMED. Por último, cada LMED definido debe ser documentado de manera adecuada para facilitar su comprensión por parte de expertos y desarrolladores. En su experiencia aplicando el desarrollo documental, los autores de este capítulo han comprobado que, mientras que SGML permite una mayor economía en el marcado que XML, los expertos en el dominio tienen menos dificultades comprendiendo la sintaxis más simplificada de XML. Así mismo, estos autores han basado satisfactoriamente la definición de sus LMEDs en el formalismo de las DTDs, primero de SGML y, posteriormente, de XML. Aunque las DTDs son más simples que otros lenguajes de esquema, en base a las experiencias con desarrolladores y expertos en el dominio, se ha comprobado que dicho formalismo es más sencillo de utilizar por los primeros en la formalización de LMEDs, y es más fácilmente comprendido por los segundos en el uso de los lenguajes así formalizados. Este hecho ha llevado al desarrollo de la técnica PADDS para la definición incremental de LMEDs basada en DTDs que se detalla en la sección 5.4.

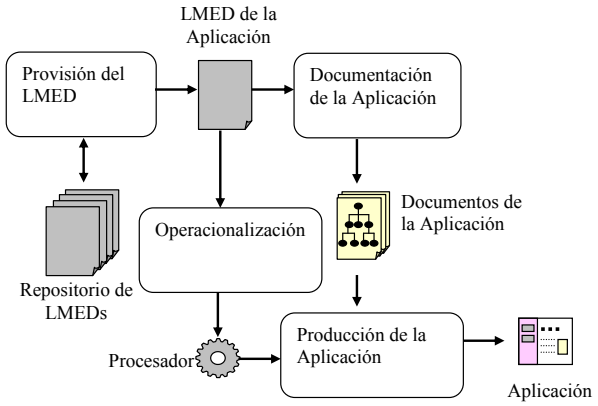


Figura 5.6. Actividades y Productos en ADDS.

Durante la actividad de **`Documentación`** la aplicación se describe mediante el conjunto de documentos de la aplicación, como ya se ha indicado. Estos documentos se marcan con el LMED de la aplicación. Como ya se ha argumentado anteriormente, la actividad de documentación se ve facilitada por la naturaleza documental de la representación elegida, así como por la naturaleza descriptiva del LMED y el carácter orientado a la aplicación de dicho lenguaje. De esta forma, los expertos en el domino son capaces de comprender dicha representación, así como de producirla y mantenerla. Para ello pueden valerse de editores orientados al metalenguaje de marcado utilizado en la definición del LMED (v.g. editores para XML, como XML Spy [37]), de editores específicos para el LMED en sí, o incluso de llevar a cabo

dicha tarea sin ningún soporte de edición específico [8], gracias a la satisfacción del requisito de comprensibilidad en la formulación del LMED,

Por último, el objetivo de la actividad de **`Operacionalización`** es proporcionar un procesador apropiado para el LMED de la aplicación, mientras que el de la actividad de **`Producción de la Aplicación`** es generar la aplicación rica en contenidos procesando sus documentos con dicho procesador.

5.3.2 La perspectiva de secuenciación de las actividades

El desarrollo de las aplicaciones ricas en contenidos es iterativo e incremental por naturaleza, tal y como se ha indicado en la sección 5.2. Durante el ciclo de vida de la aplicación es posible añadir nuevos contenidos, refinar contenidos ya existentes, y ajustar de manera apropiada las características operacionales de la misma. En cada una de estas iteraciones es posible descubrir nuevas necesidades de marcado no contempladas por el LMED de la aplicación. De esta forma, este LMED debe evolucionar para acomodar dichas necesidades. Por su parte, dicha evolución debe también reflejarse a nivel operacional en una evolución del procesador asociado con dicho lenguaje.

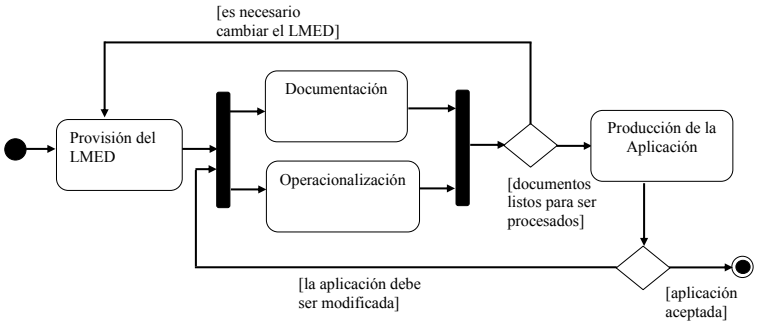


Figura 5.7. Secuenciación de las actividades en ADDS.

La Figura 5.7 muestra la secuenciación de las actividades en ADDS. Dicha secuenciación acomoda el desarrollo documental al carácter iterativo-incremental de la producción y el mantenimiento de las aplicaciones ricas en contenidos. El proceso comienza con la provisión del LMED de la aplicación. Una vez disponible dicho LMED, se proporciona la documentación inicial de la variabilidad de la aplicación, así como el procesador para dicho LMED. Esto permite producir una versión inicial de la aplicación. A partir de aquí el proceso de desarrollo procede iterativamente hasta que se alcanza una aplicación satisfactoria. Las iteraciones típicas que surgen en este proceso son de tres tipos:

- **`Iteraciones de mantenimiento`** (Figura 5.8a). Estas iteraciones suponen la producción de una versión de la aplicación, así como su evaluación por parte de los expertos. En esta evaluación, los expertos pueden descubrir distintos aspectos que deben ser mejorados o completados. Como consecuencia, los expertos

modifican y refinan los documentos de la aplicación. Como resultado se produce una aplicación mejorada. Dichas iteraciones constituyen la característica básica del desarrollo documental: los expertos en el dominio producen y refinan los documentos de la aplicación, y la aplicación en sí se produce mediante el procesamiento automático de dichos documentos. De esta forma, la mayor parte de las iteraciones en ADDS serán de este tipo.

- **`Iteraciones correctivas`** (Figura 5.8b). Durante la producción de la aplicación es posible descubrir un comportamiento no esperado en la aplicación que no puede ser atribuido a defectos en sus documentos. Normalmente esto revela un error o un aspecto que debe ser mejorado en el procesador para el LMED. De esta forma, los desarrolladores deben solucionar el problema descubierto, resolviendo el mismo en el procesador. Una vez que dicho problema se ha solucionado, la aplicación puede producirse de nuevo.

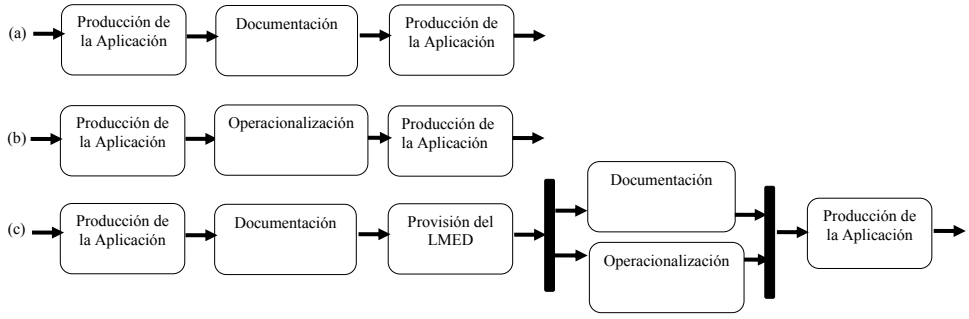


Figura 5.8. (a) Una iteración de mantenimiento; (b) una iteración correctiva; (c) una iteración evolutiva.

- **`Iteraciones evolutivas`** (Figura 5.8c). Este tipo de iteraciones suponen la evolución del LMED de la aplicación, así como del procesador asociado con el mismo. Efectivamente, durante la realización de la actividad de documentación en una iteración de mantenimiento usual es posible descubrir nuevas necesidades de marcado no contempladas por dicho LMED. Dichas necesidades se pueden deber a un refinamiento en la estructura de los documentos de la aplicación, o a la incorporación de nuevos aspectos en dichos documentos a fin de dotar a la aplicación de una nueva característica. De esta forma, es necesario extender el LMED para tener en cuenta las nuevas necesidades. Dicha extensión supone una evolución del LMED. Una vez disponible el LMED resultante, los documentos de la aplicación pueden refinarse utilizando el nuevo marcado. Así mismo, el procesador debe extenderse para tratar adecuadamente dicho marcado.

De esta forma, ADDS regula un desarrollo documental en el que no sólo los documentos de la aplicación son objeto de potencial evolución, sino que también lo son el LMED de la aplicación y su procesador asociado. La naturaleza evolutiva de los LMEDs alivia los elevados costes iniciales derivados de su formulación y su operacionalización. De hecho, tales costes pueden distribuirse a lo largo de todo el proceso de desarrollo, y amortizarse en la construcción de aplicaciones similares. Así

mismo, esta estrategia incremental también aumenta la usabilidad de los LMEDs, ya que permite evitar la inclusión prematura en el lenguaje de mecanismos descriptivos innecesariamente generales o sofisticados. No obstante, la evolución de los LMEDs y de sus procesadores asociados debe ser adecuadamente gestionada. Para tal fin es posible adoptar técnicas sistemáticas de definición de lenguajes y construcción de sus procesadores, tales como las ilustradas en [1,15,27], así como técnicas de **definición y operacionalización modular de lenguajes** [5,17,22], según las cuáles los lenguajes se definen y operacionalizan mediante la apropiada combinación de componentes reutilizables que capturan distintas características de los mismos. En esta última línea es posible encuadrar la solución PADDS, descrita en la sección 5.4, así como el modelo de operacionalización OADDS (sección 5.5).

5.3.3 La perspectiva de los actores y sus roles en las actividades

El desarrollo documental de aplicaciones ricas en contenidos involucra de manera activa tanto a expertos en el dominio como a desarrolladores, tal y como se ha discutido en la sección 5.2. De esta forma ADDS contempla ambos tipos de actores, así como su colaboración en el proceso total para producir los distintos productos contemplados en la subsección 5.3.1. De esta forma, tanto los expertos como los desarrolladores participan en cada una de las actividades del modelo, jugando distintos roles. Dicha participación se esquematiza en la Figura 5.9 y puede resumirse como sigue:

- Durante la actividad de Provisión del LMED el principal rol de los desarrolladores es la formalización del LMED mediante una gramática documental. Para ello cuentan con la ayuda de los expertos en el dominio. Obsérvese que la participación de los expertos en el dominio en esta actividad es crucial, puesto que la usabilidad del LMED finalmente producido dependerá directamente de la misma. Los expertos conocen el dominio de la aplicación, por lo que son capaces de presentar a los desarrolladores el tipo de documentos que se maneja en dicho dominio, y pueden asistirlos en la identificación de la estructura de estos documentos, así como en la elección del vocabulario de marcado más apropiado para representar dicha estructura.
- Un escenario análogo surge durante la actividad de Operacionalización. En este caso los desarrolladores construyen un procesador para el LMED, es decir, formalizan y añaden un significado operacional adecuado al marcado descriptivo. Dicho procesador involucrará operaciones relativas al dominio de la aplicación cuya naturaleza puede ser clarificada por los expertos en el dominio. De esta forma, la participación de dichos expertos en la operacionalización es igualmente crítica.

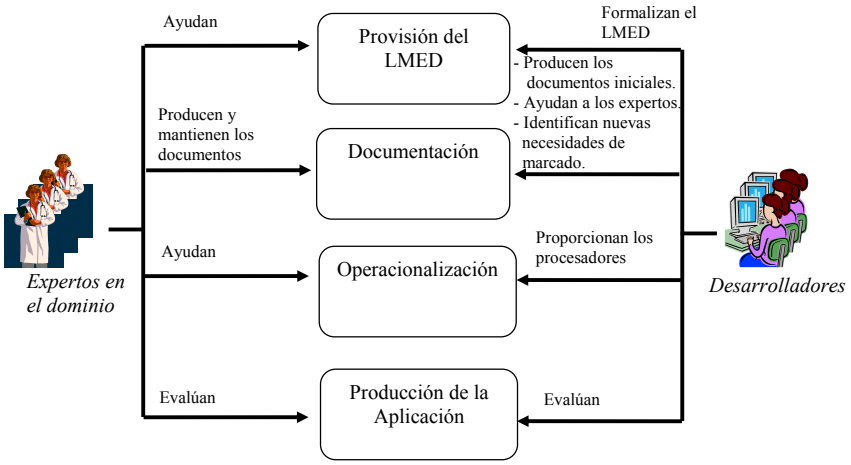


Figura 5.9. Actores en ADDS y sus roles en las actividades.

- Durante la actividad de Documentación los expertos en el dominio y los desarrolladores colaboran para describir y representar todas las características de la aplicación en los documentos marcados asociados con la misma. Los desarrolladores pueden producir y marcar versiones iniciales de dichos documentos, que pueden ser entonces entendidos, refinados y mantenidos por los expertos en el dominio. Así mismo, los desarrolladores pueden asistir a dichos expertos en la comprensión y en el uso del LMED. Por último, también pueden detectar las nuevas necesidades expresivas manifestadas por los expertos y no cubiertas por dicho LMED. La consecuencia de dicha detección será la realización de una nueva iteración evolutiva.
- Durante la producción de la aplicación tanto los expertos como los desarrolladores evalúan la aplicación producida. Mientras que los expertos pueden validar la adecuación de la aplicación con respecto a sus requisitos funcionales, los desarrolladores pueden juzgar la adecuación de la misma con respecto al resto de los requisitos no funcionales, descubriendo, por ejemplo, errores u otro tipo de problemas, como un bajo rendimiento o un tiempo de respuesta demasiado grande. Los defectos descubiertos por los expertos en el dominio desencadenan normalmente nuevas iteraciones de mantenimiento, mientras que aquellos descubiertos por los desarrolladores dan lugar a nuevas iteraciones correctivas.

5.3.4 Un ejemplo de desarrollo ADDS

En esta subsección se ilustra el desarrollo ADDS de la aplicación para la recomendación de rutas en la red de metro de Madrid a la que ya se ha hecho alusión en la subsección 5.2.4. La Figura 5.10 muestra las actividades y productos ADDS involucrados en el desarrollo de dicha aplicación. De esta forma, la actividad de Provisión del LMED proporciona un LMED para la aplicación de recomendación de

rutas del tipo sugerido en la Figura 5.5a. Nótese que en este LMED es posible identificar varios sublenguajes potencialmente reutilizables (v.g. el LMED para la descripción de redes de metro, y, dentro de éste, LMEDs más simples para la descripción de la estructura de la red, así como para la descripción de su dinámica). Todos estos sublenguajes pueden almacenarse en un repositorio de LMEDs y reutilizarse en el desarrollo de otras aplicaciones en este dominio. Por su parte, la actividad de Documentación produce un documento de la aplicación como el esbozado en la Figura 5.5b. Durante la actividad de Operacionalización se construye un procesador para el citado LMED, y la aplicación de recomendación de rutas se produce durante la actividad de Producción de la Aplicación procesando el documento de la aplicación con dicho procesador.

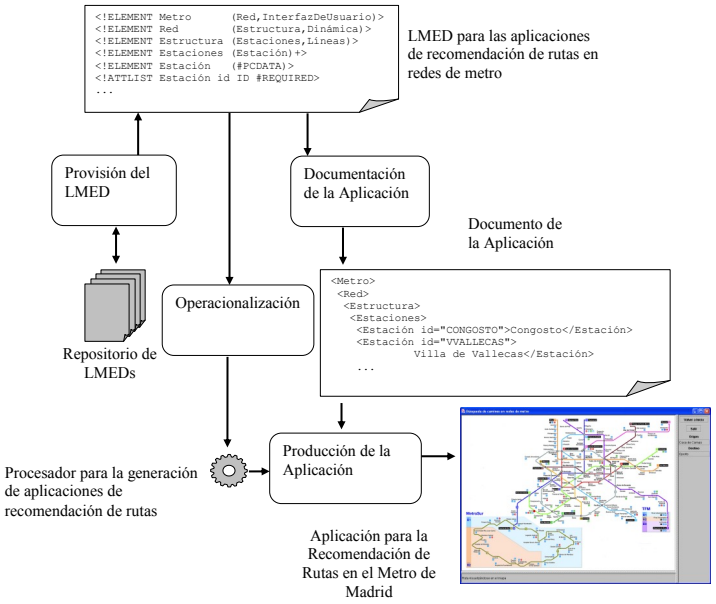


Figura 5.10. Productos y actividades en el desarrollo ADDS de una aplicación para la recomendación de rutas en la red de metro de Madrid

Nótese que el desarrollo de esta aplicación puede hacer uso de iteraciones de los tres tipos contemplados en la subsección 5.3.2:

- El desarrollo de esta aplicación puede involucrar múltiples iteraciones de mantenimiento. De esta forma, en una primera iteración es posible documentar un subconjunto preliminar de la red de metro a fin de proporcionar un primer prototipo operativo de la aplicación final. Esta documentación puede completarse en sucesivas iteraciones a fin de tratar con la red completa del metro de Madrid. Finalmente, es posible fijar las características operacionales de la aplicación, en este caso, las propiedades de su interfaz de usuario. Posteriormente podría asistirse a nuevas iteraciones de mantenimiento como respuesta a cambios en la red real, para añadir, por ejemplo, nuevas estaciones y/o nuevas líneas.

- Así mismo, conforme el tamaño de la red crece cabe la posibilidad de acusar tiempos de respuesta extremadamente largos, síntoma del empleo de una estrategia de búsqueda de rutas poco eficiente o inadecuada. Como consecuencia se debería iniciar una iteración correctiva a fin de mejorar dicha estrategia.
- Por último, el LMED en este dominio de aplicación puede evolucionar para incluir nuevos elementos estructurales en la red (v.g. corredores) junto con su dinámica asociada. Otro ejemplo de evolución es la inclusión de diferentes estilos de interfaz de usuario (v.g. evolución desde una interfaz de usuario basada en consola a una interfaz en modo gráfico).

Finalmente, y tal y como ya se ha indicado anteriormente, en este dominio de aplicación los expertos en el dominio estarán representados fundamentalmente por expertos en la gestión de redes de transporte, capaces de proporcionar todos los contenidos relativos a las redes de metro objeto de cada aplicación particular.

5.4 Formulación incremental de lenguajes de marcado específicos del dominio

La factibilidad práctica del desarrollo documental depende en gran medida de la disposición de mecanismos que permitan definir y operacionalizar incrementalmente LMEDs, tal y como se ha indicado en la sección anterior. En esta sección se describe y ejemplifica la técnica de Provisión de LMEDs en ADDS (PADDS), una técnica desarrollada por los autores de este capítulo para facilitar la definición incremental de LMEDs mediante DTDs XML en el contexto de ADDS. La subsección 5.4.1 proporciona una visión general de la técnica. La subsección 5.4.2 proporciona un ejemplo de su uso.

5.4.1 La técnica PADDS.

PADDS es una técnica de definición incremental de LMEDs en ADDS que los autores de este capítulo han desarrollado para facilitar dicha definición mediante el uso de DTDs XML. La Figura 5.11 esquematiza la estructura de la definición de un LMED en PADDS.

El principal concepto introducido por PADDS es el de **`módulo lingüístico`**. Intuitivamente, un módulo lingüístico es una caracterización gramatical de una parte del LMED finalmente definido. En este módulo cada categoría sintáctica se representa como un **`tipo de elemento`** y tiene asociado un **`modelo de contenidos`** y una lista de **`atributos`**. Los LMEDs en ADDS se definen incrementalmente formulando módulos lingüísticos más complejos mediante la combinación y extensión de otros más simples. Una vez que se dispone del conjunto de módulos lingüísticos asociados con un LMED, dicho LMED debe describirse mediante una gramática documental. Dicha gramática se obtiene siguiendo una **`especificación de realización gramatical`**. El cometido de dicha especificación es doble. Por una parte, la especificación de realización se utiliza con el fin de resolver conflictos entre los módulos lingüísticos (v.g. conflictos de nombres,

tales como la existencia de dos tipos de elemento conceptualmente distintos con el mismo nombre). Por otra parte, esta especificación permite adaptar el vocabulario de marcado de los módulos a distintos contextos de uso (v.g. diferentes idiomas). De igual forma es posible proporcionar especificaciones de realización alternativas con el fin de obtener gramáticas documentales alternativas para el mismo LMED, tal y como se sugiere en la Figura 5.11. Dichas gramáticas generan lenguajes de marcado isomorfos entre sí, que se diferencian únicamente en el vocabulario de marcas concreto utilizado.

De esta forma, la definición incremental de LMEDs depende de la posibilidad de combinar y extender módulos lingüísticos. Para facilitar estos procesos PADDS introduce tres mecanismos básicos:

- El modelo de contenidos de un tipo de elemento puede referirse a definiciones que residen en otros módulos. En términos gramaticales, este hecho equivale a la combinación de gramáticas preexistentes mediante la adición de nuevas producciones.
- Es posible disponer de módulos paramétricos. Dichos módulos incluyen un conjunto de **`parámetros`** que juegan un papel similar a las entidades tipo parámetro en las DTDs SGML o XML [11,38]. Dichos parámetros se pueden especializar con tipos de elementos definidos en el mismo o en otros módulos. De esta forma, los parámetros pueden interpretarse gramaticalmente como no terminales que no han sido totalmente definidos. Dichos no terminales dan lugar a gramáticas abiertas que pueden posteriormente extenderse mediante la adición de nuevas producciones para los mismos.

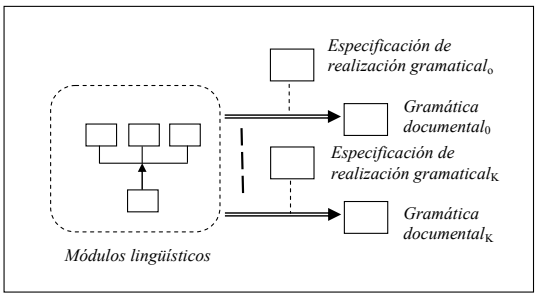


Figura 5.11. Estructura de la definición de un LMED de acuerdo con PADDS

- Por último, en un módulo es posible añadir nuevos atributos a los tipos de elementos definidos en otros módulos.

Los autores de este capítulo han llevado a cabo una implementación de PADDS totalmente basada en el metalenguaje de marcado XML. Así mismo, se ha utilizado el formalismo de las DTDs XML para describir tanto los aspectos gramaticales de los módulos lingüísticos como las gramáticas documentales finalmente producidas. Los motivos de haber elegido el formalismo de las DTDs en lugar de otros lenguajes de esquema más potentes son los ya argumentados en la sección 5.3: mayor simplicidad de uso por parte de los desarrolladores y mejor comprensibilidad por parte de los expertos en el dominio. En cualquier caso, siempre es posible la implementación de PADDS utilizando otro lenguaje de esquema (v.g. XML Schema [38]).

5.4.2 Un ejemplo de formulación incremental

La Figura 5.12 esquematiza los diferentes módulos lingüísticos que intervienen en la definición del LMED para las aplicaciones de recomendación de rutas en redes de metro. De esta forma, dicha definición involucra módulos para el marcado de las redes de metro, así como módulos para el marcado de las características más relevantes de la interfaz de usuario. Todos estos módulos pueden combinarse para producir una DTD apropiada para dicho LMED, como la esbozada en la Figura 5.5a. Para ello se sigue una especificación de realización adecuada. De hecho, es posible seguir distintas especificaciones de realización para obtener DTDs alternativas. Como un ejemplo simple, pero no por ello menos útil, la Figura 5.12 sugiere el uso de una especificación de realización que produce nombres de etiquetas en inglés, y otra que produce nombres de etiquetas en español (al estilo de los mostrados en la Figura 5.5a).

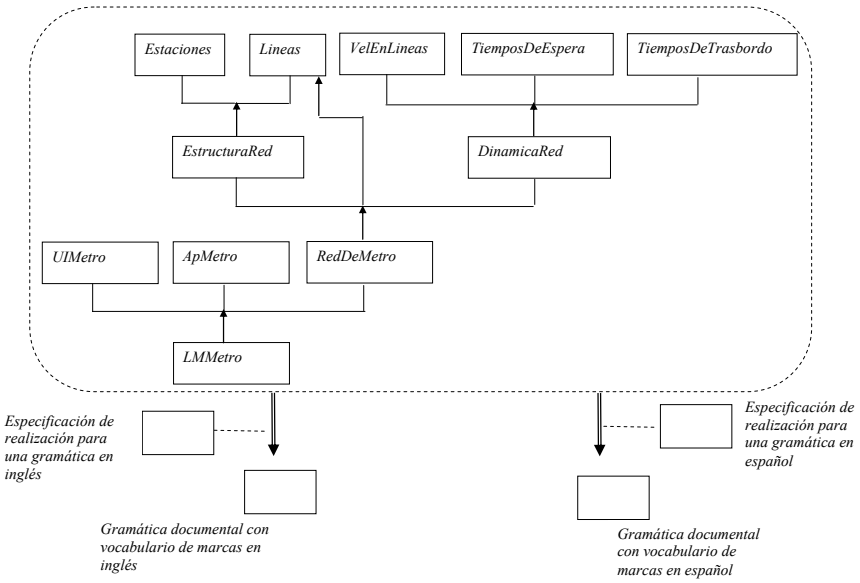


Figura 5.12. Estructura de la definición PADDS del LMED para las aplicaciones de búsqueda de rutas en redes de metro. La existencia de una flecha que parte de un módulo m_o y apunta a un módulo m_1 indica que desde m_o se refieren definiciones contenidas en m_1.

La Figura 5.13 muestra la definición de alguno de estos módulos, siguiendo la notación descrita en [28]. La Figura 5.13a muestra la definición del módulo `Lineas` para el marcado de la estructura de las líneas. Dicho módulo es un módulo básico, ya que las definiciones que incluye su DTD no hacen referencia a definiciones en otros módulos. Por su parte, la Figura 5.13b muestra la definición del módulo `RedDeMetro` para el marcado de las redes. Nótese que dicho módulo hace referencia a las definiciones en los módulos lingüísticos `EstructuraRed` y `DinamicaRed`. Para ello las definiciones referidas se cualifican con el nombre del

módulo donde han sido definidas (v.g. `EstructuraRed.Estructura`). Así mismo, en dicho módulo se extiende la lista de atributos del tipo de elemento `Líneas` en el módulo `Lineas` con un atributo `id`, lo que permite la referencia de las líneas desde el marcado de la dinámica de la red. La Figura 5.13c muestra la definición del módulo `ApMetro`. Dicho módulo incluye parámetros para referir el tipo de documentación de la red (parámetro `RED`), así como el tipo de documentación de las características operacionales relativas a la interfaz de usuario (parámetro `UI`). El módulo lingüístico `LMMETRO` mostrado en la Figura 5.13d especializa de forma adecuada dichos parámetros en términos de `RedDeMetro` y de `UIMetro`. Nótese que dicho módulo carece de gramática, puesto que su único cometido es el combinar apropiadamente `ApMetro`, `RedDeMetro` y `UIMetro`.

(a)

```
Módulo: Lineas
Gramática:
<!ELEMENT Líneas (Encabezado?,TítuloLíneas?,
                  TítuloTramos?,(Línea)+)>
<!ELEMENT Encabezado (#PCDATA)>
<!ELEMENT TítuloLíneas (#PCDATA)>
<!ELEMENT TítuloTramos (#PCDATA)>
<!ELEMENT Línea (Nombre,Tramo+)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Tramo EMPTY>
<!ATTLIST Tramo origen IDREF  #REQUIRED
                destino IDREF #REQUIRED
                longitud NMTOKEN #REQUIRED>
```

(b)

```
Módulo: RedDeMetro
Gramática:
<!ELEMENT Red (Encabezado?,EstructuraRed.Estructura,
               DinamicaRed.Dinámica )>
<!ELEMENT Encabezado (#PCDATA)>
<!ATTLIST Lineas.Línea id ID #REQUIRED>
```

(c)

```
Módulo: ApMetro
Parámetros: RED,UI
Gramática:
<!ELEMENT Metro (Encabezado?,RED,UI)>
<!ELEMENT Encabezado (#PCDATA)>
```

(d)

```
Módulo: LMMETRO
Especializaciones:
ApMetro.RED ⊇ {RedDeMetro.Red}
ApMetro.UI ⊇ {UIGrafica.InterfazDeUsuario}
```

Figura 5.13. Definiciones de algunos de los módulos lingüísticos involucrados en el LMED para la aplicación de recomendación de rutas en redes de metro.

```
...
Módulo: Lineas
  Map: Líneas  → Lines
       Encabezado → LinesHeader
       TítuloLíneas → LinesTitle
       TítuloTramos → LinksTitle
...
Módulo: RedDeMetro
  Map: Red → Network
       Encabezado  → NetworkHeader
...
Módulo: ApMetro
  Map: Metro → Subway
       Encabezado  → SubwayHeader
...
```

Figura 5.14. Parte de la especificación de realización para una gramática en inglés del LMED de las aplicaciones de recomendación de rutas en redes de metro.

Para finalizar, la Figura 5.14 muestra parte de la especificación de realización gramatical para este LMED. La notación seguida consiste en un conjunto de reglas de renombrado asociadas con cada módulo, y también se describe con detalle en [28]. La

DTD resultante exhibirá un vocabulario de marcado en inglés. Nótese que la especificación permite resolver conflictos de nombres entre los distintos módulos (v.g. el conflicto derivado de que, tanto en `Lineas`, como en `RedDeMetro` y en `ApMetro` se incluya un tipo de elemento llamado `Encabezado`).

5.5 Operacionalización incremental de lenguajes de marcado específicos del dominio

El desarrollo ADDS de una aplicación rica en contenidos necesita un procesador apropiado para el LMED de la aplicación. Dado que los LMEDs en ADDS se definen incrementalmente, es preciso también disponer de mecanismos que permiten la construcción incremental de sus procesadores. La provisión incremental de procesadores para LMEDs es un problema substancialmente más difícil que la definición incremental de dichos lenguajes. Efectivamente, dicha definición trata con aspectos sintácticos y puede abordarse de manera satisfactoria utilizando metalenguajes de marcado estándar y formalismos de descripción gramatical declarativos, tal y como se ha discutido en la sección anterior. Por su parte, la operacionalización incremental trata con aspectos semánticos, lo que constituye una fuente de complejidad adicional. Para abordar el problema de operacionalización en ADDS se ha diseñado el modelo de Operacionalización en ADDS (OADDS), un modelo conceptual que regula el desarrollo incremental de procesadores para LMEDs. OADDS se basa en las ideas clásicas de traducción dirigida por la sintaxis [1], así como en enfoques más específicos para el desarrollo modular de procesadores de lenguajes [5,17,22]. Esta sección detalla OADDS. La subsección 5.5.1 proporciona una visión general del modelo. Por su parte, la subsección 5.5.2 esboza un ejemplo de operacionalización en ADDS.

5.5.1 El modelo de operacionalización OADDS

OADDS realiza la actividad de operacionalización en ADDS, regulando la construcción de los procesadores para los LMEDs de la aplicación. Dichos procesadores permitirán procesar los documentos de las aplicaciones a fin de producir dichas aplicaciones, tal y como ya se ha indicado anteriormente en este capítulo.

Los procesadores en OADDS se construyen mediante la combinación de un conjunto apropiado de **`módulos de operacionalización`** que dotan de significados operacionales adecuados a los módulos lingüísticos utilizados en la definición de los LMEDs. De esta forma, dichos procesadores evolucionan conforme evolucionan los lenguajes procesados, ya que la inclusión de nuevos módulos lingüísticos supone la inclusión de nuevos módulos de operacionalización.

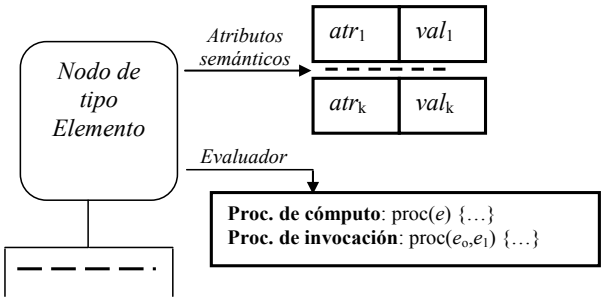


Figura 5.15. Operacionalización de elementos en OADDS.

Los módulos de operacionalización se utilizan para operacionalizar árboles documentales. Para ello se aplican sobre los nodos de tipo elemento, de ahora en adelante simplemente "elementos", de dichos árboles, y permiten decorar los mismos con un conjunto de **`atributos semánticos`** y con un **`evaluador`**, tal y como se esquematiza en la Figura 5.15. El evaluador asignado a un elemento contiene el código necesario para computar los valores de los atributos semánticos de dicho elemento. Dicho código se organiza a su vez en términos de los dos siguientes procedimientos:

- Un **`procedimiento de cómputo`** que expresa la forma de calcular los valores de los atributos semánticos en sí.
- Un **`procedimiento de invocación`**, que se utiliza en el procedimiento de cómputo para invocar los procedimientos de cómputo asociados con otros evaluadores en la vecindad del elemento al que está asignado el evaluador.

Esta estructura facilita la provisión incremental de evaluadores ya que cada uno de sus procedimientos constituyentes se puede extender de manera independiente. Así, por ejemplo, el procedimiento de invocación puede extenderse para propagar el valor de un nuevo atributo entre el elemento y los elementos en su vecindad sin necesidad de cambiar el correspondiente procedimiento de cómputo.

Los módulos de operacionalización se estructuran a su vez como un conjunto de **`operacionalizadores`** y una **`asignación operacional`**:

- Los operacionalizadores son procedimientos que capturan la manera de decorar elementos con nuevos atributos semánticos, así como producen los evaluadores que deben ser utilizados con dichos elementos.
- Las asignaciones operacionales, por su parte, son procedimientos que determinan el operacionalizador más apropiado a aplicar sobre cada tipo de elemento.

El desarrollo incremental de los módulos de operacionalización se ve facilitado por el patrón de operacionalización de los elementos en OADDS, que permite añadir nuevos atributos semánticos a los mismos conforme estos se precisan, así como extender los evaluadores utilizados en el cómputo de dichos atributos. Dicho desarrollo depende, a su vez, del desarrollo incremental de operacionalizadores y asignaciones operacionales. Esto implica que:

- Los operacionalizadores pueden invocar estáticamente otros operacionalizadores, o bien pueden mantenerse paramétricos en un evaluador, es decir, pueden tomar el evaluador que debe ser extendido como entrada adicional. En este caso, dichos operacionalizadores se aplicarán dinámicamente en las asignaciones

operacionales a fin de extender los evaluadores producidos por otros operacionalizadores.
- Por su parte, el desarrollo incremental de las asignaciones operacionales se lleva a cabo mediante la invocación de otras asignaciones operacionales previamente formuladas.

Para finalizar es importante resaltar que OADDS es un modelo conceptual, y, por tanto, independiente de tecnologías de implementación concretas. No obstante, debido a sus características, el modelo puede implementarse de manera apropiada como un marco orientado a objetos. La principal ventaja de dicha implementación es el propugnar su integración con los principales marcos de aplicación para el procesamiento documental, marcos que se formulan en su inmensa mayoría en el paradigma orientado a objetos [2]. Los detalles de la implementación de OADDS como un marco orientado a objetos pueden encontrarse en [28]. En [34] se describe una versión preliminar de dicho marco.

5.5.2 Un ejemplo de operacionalización en ADDS

La Figura 5.16a muestra el módulo de operacionalización `ConstrucciónRed` que añade significado operacional al módulo lingüístico `RedDeMetro` mostrado en la Figura 5.13b. Este módulo de operacionalización incluye:
- Un operacionalizador para operacionalizar elementos de tipo `Red`. Dicho operacionalizador, `OpRed`, asocia un atributo semántico `digrafo` con dichos elementos, que contendrá la representación de la red como un grafo dirigido valorado. El procedimiento de cómputo del evaluador devuelto construye dicho grafo a partir de la información estructural y dinámica de la red, obtenida de los atributos semánticos apropiados en los nodos hijo. Por su parte, el procedimiento de invocación se limita a realizar la invocación sin llevar a cabo ningún otro tipo de acción adicional.
- Una asignación operacional que operacionaliza los elementos de tipo `Red` con dicho operacionalizador.

La Figura 5.16b, por su parte, muestra el resultado de operacionalizar un elemento de tipo `Red` con dicho módulo de operacionalización.

A fin de ilustrar las características de operacionalización incremental de OADDS considérese la introducción de una lista de estaciones cerradas (v.g. por motivo de obras). Dicha lista afectará a la representación de la estructura y de la dinámica de la red y, por tanto, deberá ser propagada a través de los elementos de tipo `Red` a los subárboles asociados con la documentación de dichos aspectos. Esta nueva característica puede introducirse incluyendo un nuevo módulo de operacionalización, `PropagaciónEstacionesCerradas`, que extiende a `ConstrucciónRed`. La Figura 5.17a muestra la definición de dicho módulo, que incluye los siguientes componentes:
- El operacionalizador `OpPropEstaciones`. Dicho operacionalizador acepta como entradas, aparte del elemento a operacionalizar, un evaluador a extender. El operacionalizador añade un atributo `cerradas` al elemento, y devuelve un evaluador cuyo procedimiento de cómputo preserva el del evaluador recibido como entrada, y cuyo procedimiento de invocación extiende al correspondiente

procedimiento en dicho evaluador. La extensión introduce la propagación del valor para el nuevo atributo semántico añadido antes de llevar a cabo la invocación.

- Una asignación operacional que aplica la de `ConstrucciónRed` y, seguidamente, el operacionalizador `OpPropEstaciones` a fin de incorporar la nueva característica a la operacionalización.

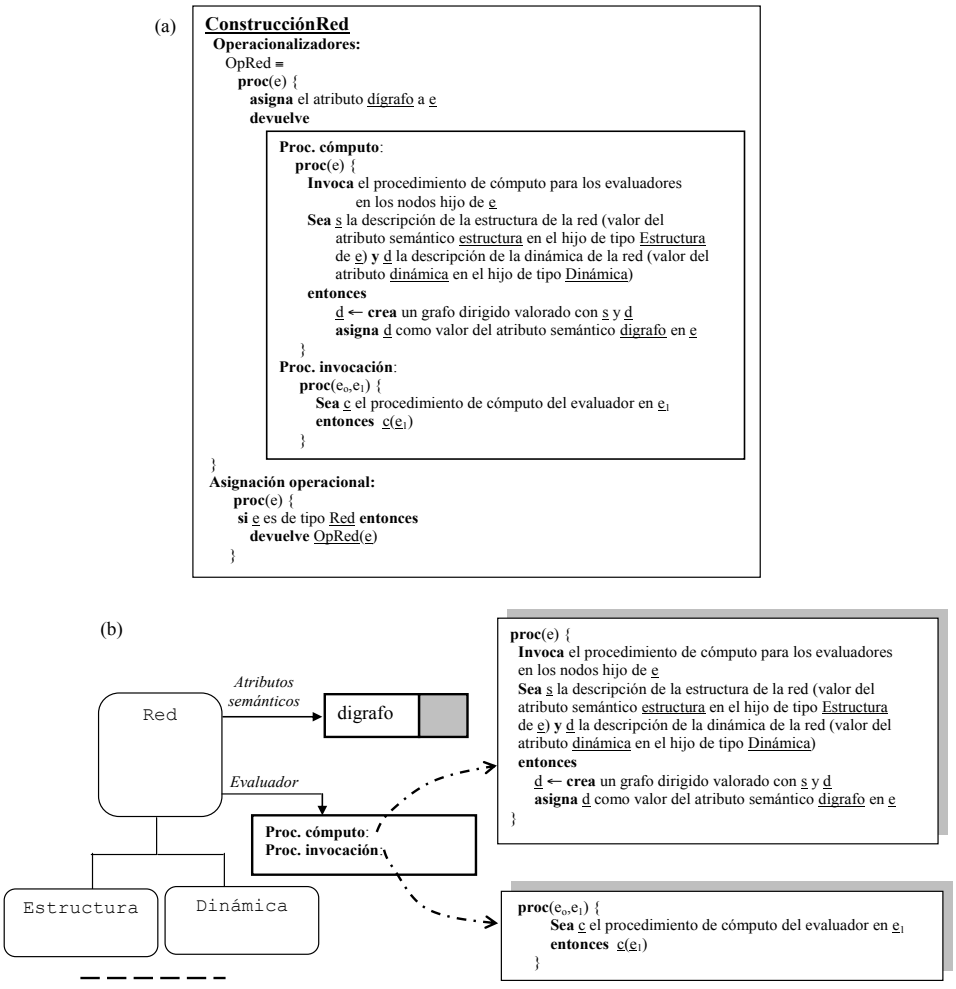


Figura 5.16. (a) Módulo de operacionalización `ConstrucciónRed`, (b) resultado de una operacionalización llevada a cabo con el módulo introducido en (a).

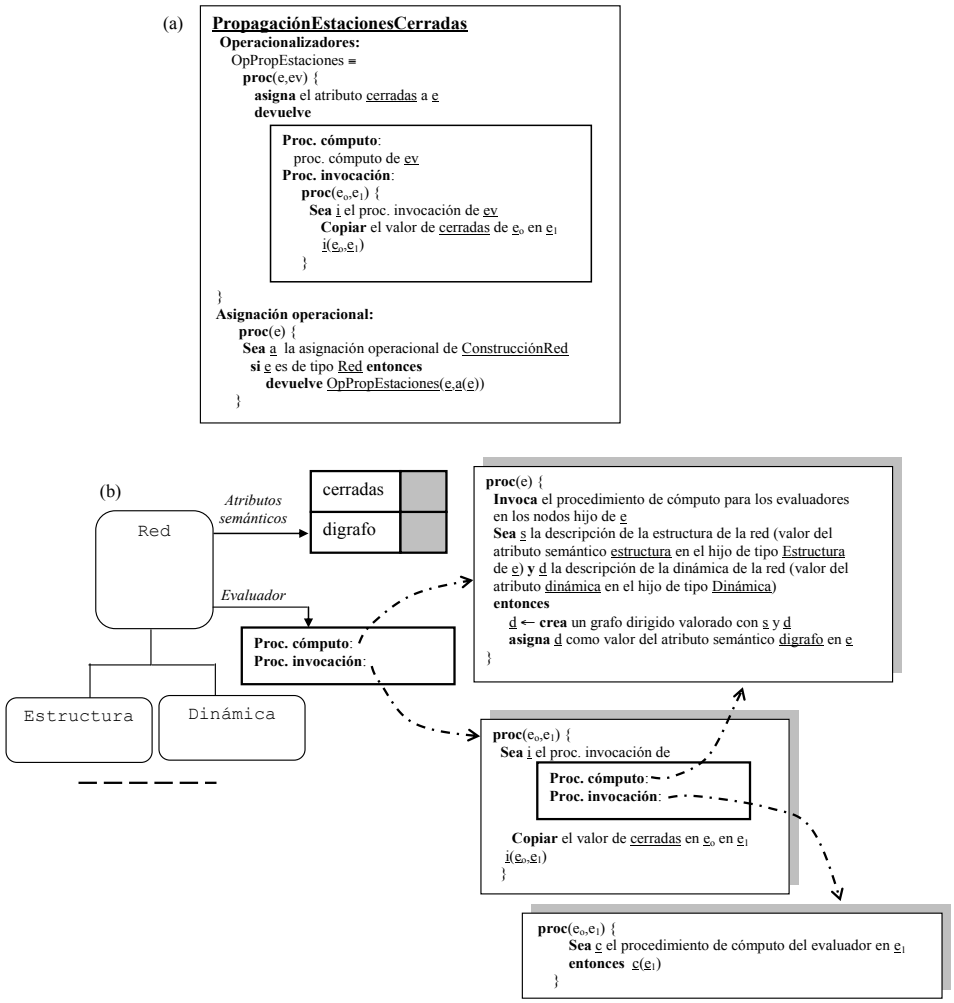


Figura 5.17. (a) Módulo de operacionalización `PropagaciónDeEstacionesCerradas`, (b) resultado de una operacionalización llevada a cabo con el módulo introducido en (a).

La Figura 5.17b muestra el resultado de operacionalizar un elemento de tipo `Red` con el nuevo módulo de operacionalización. Nótese que la operacionalización obtenida es una extensión de la mostrada en la Figura 5.16b, dónde se ha añadido un nuevo atributo semántico y dónde el procedimiento de invocación extiende al de la Figura 5.16b para propagar el valor de dicho atributo al elemento del evaluador invocado.

5.6 Trabajo Relacionado

HyTime [14,18], una aplicación SGML para la descripción de hipermedias, demostró que, en algunos dominios, los lenguajes de marcado descriptivo se podían utilizar para describir aplicaciones en términos de documentos, y que las aplicaciones descritas podían generarse mediante el procesamiento de dichos documentos. XML y sus tecnologías relacionadas han generalizado el uso de los lenguajes de marcado descriptivo como una manera estándar de intercambio de información entre aplicaciones y para otros muchos usos. Mientras que en la mayor parte de estos enfoques los lenguajes de marcado están dados a priori, desde el punto de vista del desarrollo documental se adopta una aproximación más dinámica, según la cuál los lenguajes de marcado evolucionan de acuerdo con las necesidades de marcado de los expertos en el dominio y de los desarrolladores.

El desarrollo documental comparte algunas características con la **`programación literaria`** [4,19]. El paradigma de programación literaria, propuesto originalmente en [19], mejora la comprensibilidad de las aplicaciones mediante la identificación de las mismas con su documentación. En programación literaria se lleva a cabo una representación hipertextual del código del programa, a fin de ajustar el mismo a la estructura retórica de su documentación, y dicha representación se intercala entonces con dicha documentación. El resultado, que en [19] se denomina una *web*, es una narración del programa, en la misma forma en que el programa podría presentarse en un libro de texto. Estos documentos se marcan a fin de permitir tanto el ensamblaje de programas compilables, en un proceso denominado *tangling,* como la producción de impresiones con la documentación, en un proceso que se denomina *webbing*. Las ideas descritas en este capítulo se diferencian de las de programación literaria, ya que el desarrollo documental propugna únicamente la documentación y el marcado de alto nivel de los factores de variabilidad de las aplicaciones, pero no del código de los programas que implementan dichas aplicaciones. El código en sí está implícitamente contenido en el procesador para el LMED utilizado en el marcado, de la misma forma en que el código ensamblador de los programas en el lenguaje de alto nivel compilado por un compilador puede concebirse contenido en dicho compilador. Por este motivo el desarrollo documental propugna la formulación de un LMED para cada familia de aplicaciones en lugar de utilizar un lenguaje de marcado fijo, como ocurre en programación literaria.

El trabajo descrito en este capítulo también comparte muchas características con la aproximación al desarrollo del software basada en **`lenguajes específicos del dominio`** (LEDs) [13,35]. En [10] se presenta un trabajo pionero en la aplicación de SGML/XML a la definición de LEDs. En [36] se estudian las relaciones existentes entre los lenguajes de marcado y la aproximación basada en LEDs. No obstante, aunque estos trabajos reconocen el potencial de los metalenguajes de marcado como vehículo de definición de LEDs, dicho reconocimiento se realiza concibiendo los mismos como medios de formalización de sintaxis abstractas, en lugar de enfatizar su uso como (meta) lenguajes de marcado descriptivo.

Jargons [23,24] es una aproximación similar a la propuesta en este capítulo. En *Jargons* los LEDs son formulados directamente por los expertos en el dominio, e incluso operacionalizados por dichos expertos, utilizando un lenguaje de *script*.

Mientras que este diseño de los LEDs guiado por las necesidades de autoría de los expertos es consistente con el desarrollo documental, los autores de este capítulo han constatado que no es realista asignar toda la responsabilidad de la formulación de LEDs a los expertos en el dominio, y mucho menos la responsabilidad de operacionalizar dichos lenguajes. Es por ello que en el desarrollo documental se involucra explícitamente una comunidad de desarrolladores informáticos. Así mismo, *Jargons* no contempla el problema de **modularidad semántica** en la formulación incremental de procesadores para los LEDs.

El desarrollo modular de procesadores de lenguajes se ha popularizado en el seno de la comunidad de programación funcional, donde la aproximación imperante a dicho desarrollo se basa en el uso de **mónadas** y de **transformadores de mónadas** [22]. Las ideas subyacentes a esta aproximación monádica se han aplicado, así mismo, en el contexto de la programación orientada a objetos [5], donde se utilizan ***mixins*** (clases paramétricas en sus superclases) como alternativas a los transformadores de mónadas. Así mismo, los formalismos basados en **gramáticas de atributos** [20,27] también son susceptibles de incluir mecanismos de modularidad semántica, mecanismos que normalmente implican algún tipo de patrón de propagación de atributos [17]. Estas propuestas proponen la incorporación de modularidad semántica mediante la abstracción del contexto (las entradas y salidas del procesador) y también del control (la forma de encadenar los pasos básicos de dicho procesador). OADDS está orientado a la evaluación de árboles documentales. De esta forma el modelo abstrae el contexto en términos de tablas de atributos semánticos que se pueden extender incrementalmente. Así mismo, OADDS proporciona un mecanismo limitado de abstracción de control mediante el uso de los procedimientos de invocación. Por último, OADDS se centra en los aspectos dinámicos del procesamiento. Es por ello que el énfasis del modelo se pone en dichos aspectos en lugar de en el tipado estático de los procesadores, como en otras aproximaciones [5,22].

ADDS se ha formulado y refinado a lo largo de varios años [28,29,32,33,34]. En [29,33] la aproximación se denominó **Documentos, Transformaciones y Componentes** (DTC). PADDS se ha propuesto recientemente en [28]. Los orígenes de OADDS están en [33]. En [30] se describe un primer intento de introducción de mecanismos de modularidad semántica en el modelo. En [32,34] se describen versiones preliminares del modelo, más cercanas a la presentada en este capítulo.

5.7 Conclusiones y Trabajo Futuro

Este capítulo propone un desarrollo documental de aplicaciones ricas en contenidos. De acuerdo con este desarrollo, las aplicaciones se obtienen mediante el procesamiento automático de los documentos marcados que describen las características más relevantes de las mismas. Esta aproximación se ha aplicado satisfactoriamente al desarrollo de aplicaciones educativas e hipermedia, y también al desarrollo de sistemas basados en el conocimiento. Dichas experiencias han conducido a la formulación del modelo de proceso ADDS, que sistematiza el desarrollo documental. ADDS está guiado por las necesidades de autoría de los

expertos en el dominio y los desarrolladores, por lo que los LMEDs en ADDS evolucionan de acuerdo con dichas necesidades. PADDS permite definir incrementalmente dichos LMEDs, mientras que OADDS permite construir incrementalmente sus procesadores.

El desarrollo documental facilita la producción y el mantenimiento de las aplicaciones ricas en contenidos, porque estas aplicaciones pueden describirse en términos de documentos que pueden ser entendidos y editados tanto por expertos en el dominio como por desarrolladores. La aproximación también mejora la reutilización de la información, ya que es posible utilizar estándares de marcado estándar para la producción de los documentos (v.g. SGML o XML). Esta mejora es particularmente relevante para el desarrollo de aplicaciones ricas en contenidos, donde la representación de los contenidos en formatos portables y estándares es crítica a fin de alargar sus ciclos de vida. Por último, la aproximación también mejora la reutilización del software, ya que una vez que se dispone de un procesador para un LMED en un dominio de aplicación, dicho procesador puede utilizarse sobre los documentos de todas las aplicaciones en dicho dominio.

La naturaleza evolutiva de los LMEDs en ADDS proporciona la flexibilidad requerida por el desarrollo de aplicaciones complejas. Los LMEDs pueden extenderse conforme se descubren nuevas necesidades de marcado. Esto también aumenta la usabilidad de dichos lenguajes, dado que evita la inclusión de artefactos descriptivos innecesariamente generales en los mismos. El uso de modelos para el desarrollo incremental de procesadores, como OADDS, permiten paliar el impacto operacional producido por la evolución de estos lenguajes, permitiendo la reutilización de los procesadores cuando los lenguajes procesados se extienden y/o se reutilizan en el contexto de LMEDs más complejos. Por último, esta naturaleza evolutiva ayuda a paliar los principales inconvenientes del desarrollo documental, heredados de las estrategias más generales al desarrollo del software basado en LEDs: la necesidad de formular, operacionalizar y mantener lenguajes y sus procesadores [35].

Los autores de este capítulo están actualmente trabajando en la mejora de la aplicabilidad práctica de ADDS mediante su uso en varios proyectos en el dominio de sistemas *e-learning* distribuidos en la web. Así mismo, se está trabajando en una mejor caracterización de los problemas de autoría en ADDS, no sólo en la actividad de documentación, sino también en el resto de las actividades. Por último, y como trabajo futuro, se tiene previsto llevar a cabo el ensayo de otras alternativas a PADDS y a OADDS basadas en el uso de gramáticas de atributos orientadas a objetos [27].

Agradecimientos

Este trabajo ha sido financiado a través de los proyectos CICYT TIC2001-1462, TIN2004-08367-C02-02 y TIC2002-04067-C03-02 del Ministerio de Ciencia y Tecnología.

Referencias

[1] Aho, A.; Sethi, R.; Ullman, J. D. Compilers: Principles, Techniques and Tools. Adisson-Wesley. 1986

[2] Birbeck, M et al. Professional XML 2nd Edition. WROX Press. 2001

[3] Coombs, J. H.; Renear, A. H.; DeRose, S. J. Markup Systems and the Future of

Scholarly Text Processing. Communications of the ACM 30(11). 1987
[4] Cordes,D.; Brown,M. The Literate Programming Paradigm. IEEE Computer 24(6). 52-61. 1991
[5] Duggan, D. A. Mixin-Based Semantic-Based Approach to Reusing Domain-Specific Programming Languages. ECOOP'2000. Cannes. France. 2000
[6] Fernández-Manjón, B.; Fernández-Valmayor, A.; Navarro, A. Extending Web Educational Applications via SGML Structuring and Content-based Capabilities. In Wibe,J.; Verdejo,F (Eds). The Virtual Campus. Trends for Higher Education and Training. Chapman-Hall. 1997
[7] Fernández-Manjón,B.; Fernández-Valmayor,A. Improving World Wide Web Educational Uses Promoting Hypertext and Standard General Markup Languages. Education and Information Technologies 2(3). 1997
[8] Fernández-Valmayor, A.; López Alonso, C.; Sèrè A.; Fernández-Manjón,B. Integrating an Interactive Learning Paradigm for Foreign Language Text Comprehension into a Flexible Hypermedia system. IFIP WG3.2-WG3.6 Conf. Univ. California Irvine, California, USA. 1999
[9] Fraternali, P. Tools and Approaches for Developing Data Intensive Web Applications: A Survey. ACM Computing Surveys 3. 227-263. 1999
[10] Fuchs, M. Domain Specific Languages for *ad hoc* Distributed Applications. First Conf. on Domain Specific Languages. Sta. Barbara. CA. 1997
[11] Goldfarb, C. F. The SGML Handbook. Oxford University Press. 1990
[12] Goldfarb, C.F. A Generalized Approach to Document Markup, ACM SIGPLAN Notices 16(6).68-73. 1981
[13] Hudak,P. Domain-Specific Languages. In Handbook of Programming Languages Vol. III: Little Languages and Tools, Macmillan Tech. Publishing (1998) 39-60.
[14] International Standards Organization. Hypermedia/Time-based Structuring Language (HyTime) – 2d Edition. ISO/IEC 10744. 1997
[15] Johnson,S.C. YACC-yet Another Compiler-Compiler. Computing Science Technical Report 32. AT&T Bell Laboratories. 1975
[16] Juristo,N.; Pazos,J. Towards a Joint Life Cycle for Software and Knowledge Engineering. IFIP Transactions A-27. 119-138. 1993
[17] Kastens,U.; Waite,W.M. Modularity and Reusability in Attribute Grammars. Tech. Report CS-613-92. University of Colorado. 1992
[18] Kimber, W. E. A Tutorial Introduction to SGML Architectures. ISOGEN International Corporation Workpaper. 1997
[19] Knuth,D.E. Literate Programming. The Computer Journal 27(2). 97-111. 1984
[20] Knuth,D.E. Semantics of Context-free Languages. Math. Systems Theory 2(2). 127-145. 1968
[21] Lee,D.; Chu,W.W. Comparative Analysis of Six XML Schema Languages. ACM SIGMOD Record. 29(3). 2000
[22] Liang,S.; Hudak,P.; Jones,M.P. Monad Transformers and Modular Interpreters. 22nd ACM Symp. on Principles of Programming Languages. San Francisco. CA. 1995
[23] Nakatani, L.H.; Ardis, M.A.; Olsen, R.G.; Pontrelli, P.M. Jargons for Domain Engineering. 2º Conf. for Domain Specific Languages. Austin. Texas. 1999
[24] Nakatani, L.H.; Jones, M. Jargons and Infocentrism. First ACM SIGPLAN Workshop on Domain-Specific Languages DSL'97. Paris, France. 1997
[25] Navarro A.; Fernández B.; Fernández-Valmayor A.; Sierra J.L. The PlumbingXJ Approach for Fast Prototyping of Web Applications. J. of Digital Inf. 5 (2). 2004
[26] Navarro, A.; Fernández-Valmayor, A.; Fernández-Manjón, B.; Sierra, J.L. Conceptualization prototyping and process of hypermedia applications. International Journal of Software Engineering and Knowledge Engineering, 14(6). 565-602. 2004
[27] Pakki, J. Attribute Grammar Paradigms – A High-Level Methodology in Language

Implementation. ACM Computing Surveys 27(2). 196-255. 1995
[28] Sierra, J.L. Hacia un Paradigma Documental de Desarrollo de Aplicaciones. Tesis Doctoral. Universidad Complutense de Madrid. 2004.
[29] Sierra, J.L.; Fernández-Manjón, B.; Fernández-Valmayor, A.; Navarro, A. Integration of Markup Languages, Document Transformations and Software Components in the Development of Applications: the DTC Approach. Int. Conf. on Software ICS 2000. 16th IFIP World Comp. Congress. Beijing - China. 2000
[30] Sierra, J.L.; Fernández-Manjón, B.; Fernández-Valmayor, A.; Navarro, A. An Extensible and Modular Processing Model for Document Trees. Extreme Markup Languages 2002. Montreal. Canada. 2002
[31] Sierra, J.L.; Fernández-Manjón, B.; Fernández-Valmayor, A.; Navarro, A. A Document-Oriented Approach to the Development of Knowledge-Based Systems. In Conejo,R.; Urretavizcaya, M.; Pérez-de-la-Cruz,J.L. Current Topics in Artificial Intelligence.LNAI 2040. Springer-Verlag. 2004
[32] Sierra, J.L.; Fernández-Valmayor, A.; Fernández-Manjón, B.; Navarro, A. Building Applications with Domain-Specific Markup Languages: A Systematic Approach to the Development of XML-based Software. ICWE 2003. Oviedo. 2003
[33] Sierra, J.L.; Fernández-Valmayor, A.; Fernández-Manjón, B.; Navarro, A. Operationalizing Application Descriptions with DTC: Building Applications with Generalized Markup Technologies. SEKE'01. Buenos Aires. Argentina. 2001
[34] Sierra, J.L.; Fernández-Valmayor, A.; Fernández-Manjón,B.; Navarro,A. ADDS: A Document-Oriented Approach for Application Development. Journal of Universal Computer Science 10(9) .1302-1324. 2004
[35] Van Deursen, A.; Klint, P.; Visser, J. Domain-Specific Languages: An Annotated Bibliography. ACM SIGPLAN Notices 35(6). 26-36. 2000.
[36] Wadler,P. The next 700 markup languages. Invited Talk of the 2º USENIX Conf. on Domain Specific Languages. USENIX. Austin. Texas. 1999
[37] www.altova.com/XMLSpy
[38] www.w3.org/TR

Desarrollo de Entornos Virtuales para Web

María-Isabel Sánchez-Segura[1], Angélica de Antonio[2], Gonzalo Méndez[2]

[1] Avda. de la Universidad, 30
28911 Leganés (Madrid), España
Tfno: +34 91 624 94 21
Fax: +34 91 624 91 29
Departamento de Informática. Universidad Carlos III de Madrid
misanche@inf.uc3m.es

[2] Campus de Montegancedo s/n
28660 Boadilla del Monte (Madrid), España
Tfno: +34 91 336 69 25
Fax: +34 91 336 69 17
Facultad de Informática. Universidad Politécnica de Madrid
angelica@fi.upm.es, gonzalo@gordini.ls.fi.upm.es

Resumen. Los Entornos Virtuales constituyen un tipo de sistema altamente interactivo para cuyo desarrollo se hace necesario encontrar nuevas técnicas y formalismos especialmente adaptados, ya que los métodos y técnicas clásicos de la Ingeniería del Software han demostrado empíricamente no ser suficientes. En este capítulo, tras una introducción a los entornos virtuales, sus principales características y tipologías, y una revisión de las diferentes tecnologías involucradas en su construcción, se presenta una aproximación al proceso de desarrollo para este tipo de sistemas, el marco metodológico SENDA, profundizando en el conjunto de actividades y técnicas propuestas para la fase de análisis. Por último se hace una revisión de las principales áreas en que estos sistemas están teniendo acogida.

6.1 Introducción

En 1989, Jaron Lanier propone el término de **Realidad Virtual** (RV), y se define como "una simulación interactiva que implica a todos los sentidos, generada por un ordenador, explorable, visualizable y manipulable en tiempo real, dando la sensación de presencia en el entorno". Un aspecto clave en esta definición es que la sensación –visual, auditiva, táctil... – se debe percibir como auténtica por el sujeto.

El término **Entorno Virtual** (EV) fue introducido por investigadores del MIT (*Massachusetts Institute of Technology*) a principios de los años 90 como sinónimo de Realidad Virtual, aunque posteriormente se ha venido utilizando este término para designar algo más general, incluyendo también sistemas en los que la sensación de presencia del usuario en el entorno no es tan fuerte.

En cualquier caso, lo realmente diferente en un sistema de realidad virtual o entorno virtual frente a otros tipos de sistemas interactivos es el hecho de que el usuario pasa a formar parte integrante del entorno, en lugar de interactuar con él desde fuera, y habitualmente adopta algún tipo de representación dentro del entorno que puede manipular en tiempo real. A esta representación de un usuario dentro de un entorno virtual se le ha venido llamando **Avatar**.

Un avatar puede adoptar cualquier forma, desde un simple nombre identificativo, como ocurría en los primeros y más primitivos entornos virtuales, que eran puramente textuales – los conocidos como MUDs (*Multi-User Dungeons*) – hasta un maniquí tridimensional con forma humanoide.

Lo importante de un avatar es que proporciona al usuario una identidad; puede servir a otros usuarios como indicador de presencia, posición y actividad; y es utilizado por el usuario al que representa como medio para interactuar con el resto del entorno.

Los avatares de tipo humanoide facilitan una mayor identificación del usuario con su avatar, pero, dada la mayor complejidad inherente en su construcción y animación, si no se consigue un alto grado de realismo pueden incluso llegar a entorpecer la verosimilitud o credibilidad del entorno, pudiendo ser contraproducentes. En la Figura 6.1 se aprecia el contraste entre un avatar humanoide y otro de diseño fantástico.

Una vez aclarada la terminología más básica en relación con los entornos virtuales, pasaremos a continuación a analizar sus principales características.

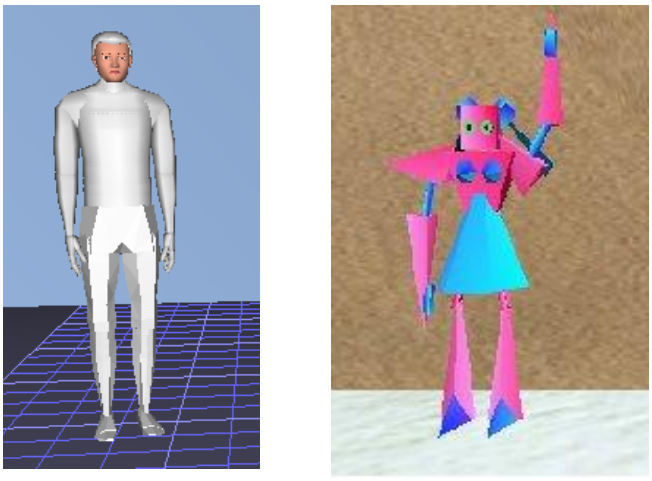


Figura 6.1. Ejemplos de avatares

Este capítulo se estructura del siguiente modo. Este primer apartado de Introducción está dedicado a describir conceptos generales relacionados con los Entornos Virtuales, y especialmente con uno de los elementos principales de estos que son los habitantes de los mismos. En el apartado 6.2 se presentan las distintas tecnologías que se pueden utilizar para desarrollar Entornos Virtuales tanto fuera como dentro de un entorno Web de desarrollo. El apartado 6.3, está dedicado a la descripción detallada del proceso de análisis que se debe llevar a cabo para desarrollar un Entorno Virtual, independientemente de la plataforma e desarrollo que se haya seleccionado. En el apartado 6.4 se recogen las principales aplicaciones de los Entornos Virtuales y por último en el apartado 6.5 se resumen las principales conclusiones que se extraen de la exposición llevada a cabo a lo largo del presente capítulo.

6.1.1 Tipos y características principales de los entornos virtuales

Los entornos virtuales, pueden ser clasificados de acuerdo a múltiples características que serán brevemente analizadas en los siguientes apartados. Es importante identificar el tipo de entorno virtual que se pretende desarrollar de acuerdo con estos parámetros, ya que diferentes variedades de entornos virtuales impondrán requisitos diferentes y exigirán tipos de diseños diferentes.

Dimensionalidad
Quizás la clasificación más inmediata de los entornos virtuales sea de acuerdo con su dimensionalidad en el espacio de presentación, encontrándonos entornos **Textuales** – también llamados Unidimensionales–, **Bidimensionales** y **Tridimensionales**. Los EVs han evolucionado hacia una mayor dimensionalidad en paralelo con la evolución de la tecnología.

Número de usuarios
Una segunda e interesante clasificación de los EVs puede realizarse en función del número de usuarios simultáneos que pueden utilizarlos, encontrándonos EVs **Mono-usuario** y EVs **Multi-usuario**, a los que también se ha dado en llamar **Entornos Virtuales Distribuidos** (DVE, *Distributed Virtual Environments*)

La historia de los DVEs se remonta a la década de los 70 y discurre por dos caminos en paralelo: el mundo de Internet, orientándose fundamentalmente a los juegos en red; y el campo militar, orientándose fundamentalmente a la simulación para el entrenamiento. A estos últimos se les llamó tradicionalmente Simulaciones Interactivas Distribuidas (DIS, *Distributed Interactive Simulation*).

Para facilitar el desarrollo de EVs multi-usuario se han creado diferentes plataformas de desarrollo, como son SPLINE [34], MASSIVE [11], NPSNET [22], o DIVE [7]. Estas plataformas facilitan al desarrollador el manejo de aspectos y problemas específicos que tienen este tipo de sistemas, como son las comunicaciones a través de la red, la sincronización entre las diferentes vistas que los usuarios tienen sobre el entorno, la propagación de eventos de unos usuarios a otros, la escalabilidad del sistema cuando crece el número de usuarios simultáneos, etc.

Grado de inmersión
Un tercer criterio por el que podemos clasificar los entornos virtuales es el grado y cualidad de la sensación de inmersión que ofrecen a sus usuarios. *Grosso modo* se suelen clasificar en **Inmersivos** y **No inmersivos**, llamándose también a estos últimos **Realidad Virtual de Escritorio** (*Desktop Virtual Reality*) por el hecho de que hacen uso de dispositivos de interacción convencionales en un ordenador de escritorio, como son monitor, teclado, ratón o joystick. Los sistemas inmersivos, por el contrario, requieren de dispositivos de interacción muy especiales y generalmente costosos. La inmersión o no-inmersión en un entorno virtual da lugar a dos experiencias fundamentalmente diferentes:

- los sistemas no inmersivos soportan la sensación de "mirar al" EV.
- los sistemas inmersivos soportan la sensación de "estar en" el EV.

Grado de interactividad
Un cuarto criterio de clasificación es el grado de interactividad que ofrecen a sus usuarios. Según este criterio se clasifican en:

- **Pasivos**: son entornos en los que podemos ver, oír, y quizás sentir lo que sucede, pero no es posible controlar lo que ocurre. Corresponde a las llamadas películas dinámicas habituales en parques de atracciones.
- **Exploratorios**: permiten al usuario desplazarse por el entorno virtual para explorarlo. Es el estadio correspondiente a los paseos arquitectónicos y a las obras de arte virtuales.
- **Interactivos**: permiten al usuario explorar y experimentar con el entorno, modificándolo.

Grado de realismo
Si nos centramos en el contenido del EV y su relación con la realidad, podemos hablar de EVs **Realistas**, cuando tratan de reproducir lo más fielmente posible algún espacio o fenómeno real; **Semi-realistas**, cuando básicamente reproducen la realidad, pero adaptándola en algún aspecto para mejorar su comprensión (v.g. podemos adaptar el tamaño en un EV que reproduce el sistema nervioso humano; o la transparencia en un EV que reproduce un motor y deseamos que se vean los componentes internos del mismo; o la escala de tiempo en un EV que nos enseña cómo transcurre la creación de una nueva estrella). Finalmente, podemos encontrarnos EVs totalmente **Fantásticos** y sin relación con la realidad.

Grado de virtualidad
También podemos clasificar los EVs según su grado de virtualidad, pudiendo encontrarnos en cualquier punto de un continuo entre dos extremos constituidos por el mundo real y un mundo totalmente virtual [25]. Aparece así la llamada **Realidad Mixta**, incluyendo sistemas de **Realidad Aumentada**, cuando es el mundo real el que se ve complementado con objetos virtuales, y sistemas de **Virtualidad Aumentada**, cuando objetos del mundo real pueblan el mundo virtual.

6.1.2 Habitantes en un entorno virtual

En un entorno virtual podemos encontrarnos dos tipos diferentes de habitantes:

- **Avatares** puros, que representan y son controlados por un usuario.
- **Agentes virtuales** autónomos, que no son controlados por un usuario sino por el propio sistema.

Más precisamente podemos definir un agente virtual como un agente software, más o menos inteligente, que está encarnado (es decir, adopta una representación corpórea, generalmente con forma humana, a la que se suele llamar *embodiment*), y que habita un entorno virtual. El hecho de que habite un EV significa que no sólo percibe observaciones de un entorno externo, sino que forma parte del mismo entorno, y debe ser capaz de desenvolverse, percibir e interactuar en él.

La información a percibir no es una representación simbólica abstracta especialmente adaptada, sino información compleja de tipo visual, auditivo, táctil, etc.

A diferencia de los agentes software inteligentes no encarnados, los humanos observarán no sólo el resultado de sus acciones sobre el entorno, sino la forma en que se producen dichas acciones, y una parte importante de sus acciones implicarán la manipulación de su propio cuerpo. En estas acciones su comportamiento debe ser verosímil para un observador humano.

El diseño de avatares y agentes virtuales comparte la necesidad de dotar a ambos de un cuerpo y ciertas posibilidades de acción, pero un agente virtual requiere además el diseño de sus capacidades perceptivas y sus capacidades de razonamiento y toma de decisiones, aspectos estos totalmente ausentes en el diseño de un avatar.

En la Figura 6.2 se pueden ver los tres componentes de un agente virtual, según un modelo general de organismo.

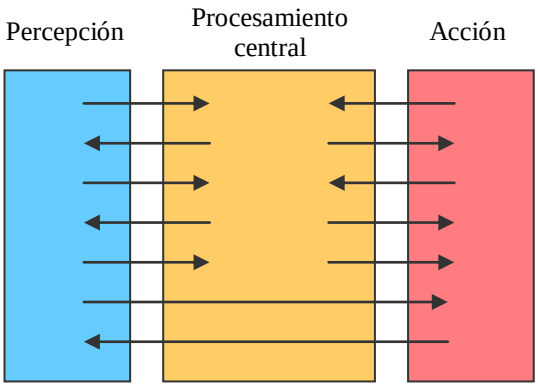


Figura 6.2. Modelo general de organismo de un agente virtual

6.2 Tecnologías para el desarrollo de entornos virtuales dentro y fuera de la web

La construcción de un entorno virtual tridimensional (3D) implica la utilización de diversas herramientas para completar todo el proceso de desarrollo. Estas herramientas, por un lado, permiten la creación de los elementos tridimensionales que estarán presentes dentro del EV, como pueden ser las estancias que conforman el mundo virtual, los avatares que representan a los usuarios y demás habitantes del mundo, y los objetos con los que es posible interactuar. Por otra parte, será necesario otro tipo de herramientas que permitan, una vez se dispone de los modelos 3D, dotarles de comportamiento y posibilitar la interacción con ellos.

Como sucede con los entornos de desarrollo tradicionales, cada vez es más habitual encontrar herramientas que incorporen todas las funcionalidades necesarias para desarrollar un EV, desde la creación de los modelos 3D hasta la interacción con dispositivos de realidad virtual, pasando por mecanismos de animación y lenguajes de programación que permiten dotar de comportamiento a los elementos presentes en el EV.

A lo largo de los siguientes apartados se van a describir distintas herramientas que se utilizan para realizar las tareas descritas, aunque no se pretende realizar un estudio detallado, pues se encuentra fuera de los objetivos del presente capítulo.

6.2.1 Herramientas para la elaboración de modelos 3D

Los objetos tridimensionales que se utilizan para construir los EVs deben cumplir unas características muy concretas, especialmente debido a la capacidad de procesamiento que exige la representación de un entorno virtual. De esta manera, aunque en la actualidad existen herramientas de modelado que permiten realizar trabajos de enorme complejidad (v.g. mediante el uso de NURBs (*NonUniform Rational B-splines*), distintos tipos de iluminación, sistemas de partículas), su potencia sólo se aprovecha para labores de cinematografía.

En lo que a la construcción de EVs se refiere, lo que se le exige a estas herramientas es poder crear los objetos 3D haciendo uso de polígonos, generalmente triángulos, colorearlos, mejorar su aspecto con el uso de texturas y, en caso necesario, construir animaciones para mostrar el comportamiento de los objetos dentro del entorno virtual. Adicionalmente, es deseable que estas herramientas guarden los objetos 3D en algún formato estándar que sea fácilmente accesible para otras herramientas, ya que, aunque es posible emplear utilidades que conviertan un formato gráfico en otro, los resultados no son siempre los deseados.

Una de las herramientas más potentes y populares es 3D Studio, que proporciona todas las funcionalidades mencionadas anteriormente. Esta herramienta es especialmente destacable porque ha dado lugar a un estándar de facto para el formato de los ficheros donde se guardan los objetos 3D, el formato 3ds, que es leído y generado por multitud de aplicaciones. Como característica adicional, también proporciona un pequeño lenguaje de *scripts* que, entre otras cosas, permite la construcción de exportadores que conviertan los objetos creados a formatos propietarios que resulten de mayor utilidad a la hora de construir el EV.

Los modelos generados por estas herramientas serán posteriormente importados desde otras aplicaciones para construir el EV propiamente dicho.

6.2.2 Herramientas gráficas

Debido sobre todo a su carácter visual, estas herramientas permiten realizar un desarrollo bastante rápido de los EVs, especialmente si lo comparamos con el tiempo de desarrollo que requiere el uso de librerías. Como contrapartida, en general constituyen un método bastante menos potente para construir los EVs, ya que no permiten tocar aspectos de bajo nivel del motor gráfico, cambiar el modo de ejecución y, en general, hacer algo que no proporcione la herramienta en sí.

WorldUp

WorldUp es una herramienta de autor para construir Entornos Virtuales. Esta herramienta permite crear los objetos 3D que van a formar el EV, y también da la posibilidad de importar los objetos diseñados con otras aplicaciones siempre que se utilicen formatos como 3ds o VRML. En el caso de crear los objetos con WorldUp, se pueden realizar tareas comunes como rotarlos, cambiarles el tamaño, el color o la textura.

Una vez cargados los objetos 3D, permite manejarlos como si se tratasen de objetos de un lenguaje de programación orientado a objetos, de manera que permite crear clases de objetos, subclases de las anteriores que heredan sus propiedades, definir atributos o asignar comportamientos como si de métodos se tratase, aunque, en este último caso, los *scripts* que contienen los comportamientos se le asignan a los objetos individualmente.

WorldUp proporciona de serie una lista de dispositivos que se pueden utilizar para navegar por los entornos virtuales creados con esta herramienta, pero también admite la utilización de módulos externos que permitan el uso de dispositivos no contemplados por WorldUp.

La manera habitual de ver un escenario es utilizar un reproductor para WorldUp. El EV se guarda en un único archivo que es abierto por el reproductor. Los reproductores pueden ser una aplicación independiente o estar contenidos en otra aplicación. También puede utilizarse un *plug-in* para los navegadores web, o un control Active-X para integrarlo con aplicaciones desarrolladas en C++ o Visual Basic.

Active Worlds

Active Worlds es una aplicación multiusuario que permite a los usuarios conectarse a una serie de entornos virtuales en 3D y conversar o comprar productos en ellos. Active Worlds proporciona servidores de mundos 3D construidos con objetos con el formato RWX (*RenderWare Script format*), formato que puede ser utilizado por varias librerías gráficas, como DirectX, y también es el formato que siguen los objetos de algunos juegos desarrollados para las videoconsolas Xbox, GameCube y Playstation-2. En estos servidores de mundos se instala un software que permite ejecutar a la vez diferentes entornos virtuales. Una vez que el EV está activo en un servidor, un usuario puede conectarse a éste por medio de un navegador y visualizar el mundo al que se ha conectado. Algunos de los EVs más grandes creados hasta la fecha tienen registrados más de 200.000 ciudadanos.

Para visualizar el EV no es necesario cargar todos sus objetos ni todos los avatares que representan al resto de los usuarios conectados, sino que simplemente se visualizan aquellos que el usuario puede ver en ese momento. Cuando se visualiza un objeto por primera vez, la información referente a él se descarga en la máquina del usuario y se almacena en ella, de manera que no es necesario volver a descargarla en posteriores ocasiones.

Uno de los aspectos más importantes de Active Worlds, y que lo diferencia de otras aplicaciones que proporcionan accesos a entornos virtuales, es la capacidad que tienen los usuarios de construir dentro del EV. Esto quiere decir que un usuario puede, en una zona vacía de un EV, añadir diferentes objetos y edificios. Para que un usuario construya en un EV debe clonar un objeto preexistente y situar el clon en un lugar que no esté ocupado. Posteriormente puede cambiar las características de este nuevo objeto. Además de construir en el EV, un usuario puede construir también el propio avatar que lo represente, en vez de elegir uno predeterminado.

Los objetos del EV deben tener el formato RWX, en el que se especifican las características del objeto. Existen diferentes formas de crear un objeto de este tipo. Una de ellas es editar el *script* del objeto y ver cómo ha quedado en un visor de objetos de este formato. Otra manera es utilizar una herramienta de modelado visual de objetos.

Por otra parte, para programar aplicaciones y hacer que utilicen Active Worlds, existen varios SDK (*Software Development Kit*) y una API (*Application Programming Interface*). La mayoría de los métodos de la API están orientados a la conexión con el servidor, la validación del usuario que se conecta al EV, a sus capacidades de teletransporte (dar un salto en el EV y aparecer en otro lugar), capacidad de volar, etc.

Active Worlds está pensado para interrelacionarse con otros usuarios y charlar, es decir, el EV es un medio y no un fin, por lo que no existen funciones complejas para la manipulación de objetos.

Jack

Es una aplicación desarrollada por la Universidad de Pennsylvania que está orientada al estudio de la ergonomía del ser humano en diferentes lugares de trabajo. Su principal valor radica en la capacidad para la simulación de cuerpos humanos con sus características biomecánicas, antropométricas y ergonómicas.

Entre las características principales de Jack se incluye un sistema para modelar figuras humanas articuladas con detección de colisiones y cinemática inversa. Estas figuras se encuentran en una librería que incorpora la propia aplicación, la cual nos permite cambiar las características de cualquiera de estas figuras para adaptarlas a nuestras necesidades. Las figuras pueden tener definidas restricciones ergonómicas, y el sistema de simulación tratará de cumplirlas con la mayor exactitud.

La aplicación también permite crear objetos simples e importar objetos creados con otras herramientas, de manera que se pueden crear escenarios complejos con bastante facilidad.

Otro punto destacable es la incorporación de los lenguajes de programación Tcl, Python y Lisp, por medio de los cuales es posible crear un entorno virtual con nuevas ventanas y controles y con las interacciones y comportamientos de los objetos que defina el desarrollador del EV.

6.2.3 Librerías de programación

Las librerías de desarrollo de gráficos tridimensionales son una potente herramienta sobre la que se apoyan las aplicaciones descritas en el apartado anterior. Aunque existe una amplia variedad de librerías, la mayoría de los EVs se desarrollan utilizando una de las dos que se van a describir a continuación: DirectX-Direct3D y OpenGL.

DirectX – Direct 3D

DirectX es una API desarrollada por Microsoft que tiene una amplia difusión para el desarrollo de juegos, aunque, cada vez más, se utiliza también para otro tipo de aplicaciones, y en concreto para el desarrollo de EVs.

Esta librería se encuentra disponible únicamente para la plataforma Windows, lo cual supone una gran desventaja frente a OpenGL, pero, a diferencia de ésta, DirectX se compone de una serie de módulos que proporcionan una funcionalidad mucho más amplia que la que provee OpenGL.

El más relacionado con el tema que nos ocupa es DirectX Graphics, que proporciona manejo de gráficos 2D (DirectDraw) y 3D (Direct3D) de manera similar a OpenGL, aunque también incluye funciones de más alto nivel para facilitar la construcción y manejo de los entornos tridimensionales.

Direct3D maneja su propio formato para los objetos 3D, el formato *X*, lo cual no impide utilizar otros formatos gráficos, eso sí, sin aprovechar las facilidades que da Direct3D para operar con el formato X. Este formato almacena la descripción geométrica de los objetos, su relación jerárquica e incluso las animaciones que se puedan haber diseñado con herramientas de modelado. Además, el SDK de DirectX incorpora *plug-ins* para 3DStudio y Maya que permiten que estas herramientas generen ya los modelos en formato *X*.

Adicionalmente, DirectX incluye otros módulos, como son DirectInput, para el manejo de distintos elementos de interacción como teclado, ratón o joystick, DirectSound, para la reproducción de música y efectos sonoros, o DirectPlay, para la gestión de conexiones de red en aplicaciones multiusuario.

OpenGL

OpenGL es una librería que sirve para el desarrollo de gráficos 2D y 3D. Creada inicialmente por Silicon Graphics en 1992, actualmente se encarga de su mantenimiento y desarrollo el *OpenGL Architecture Review Board*, un consorcio que cuenta con un amplio respaldo en el mundo de la industria.

Una de las razones por las que su uso es tan extendido se encuentra en el hecho de que es una librería que se encuentra disponible en una amplia gama de plataformas, como Windows, Linux o Irix, de manera que el código desarrollado es fácilmente portable entre ellas. Además, se puede utilizar con distintos lenguajes de programación, como C, C++, Ada, Fortran, Perl y Java, lo cual ofrece grandes atractivos a una gran variedad de desarrolladores de software.

OpenGL no proporciona funciones de alto nivel para describir modelos de objetos tridimensionales, sino que se deben construir los modelos deseados a partir de un pequeño conjunto de primitivas geométricas, como puntos, líneas y polígonos. La librería de utilidades de OpenGL (GLU, *OpenGL Utility Library*) proporciona más funcionalidades para el modelado, como superficies cuádricas y curvas y superficies NURBS. GLU es estándar en todas las implementaciones de OpenGL.

A través del trabajo con vectores y matrices, OpenGL permite realizar transformaciones de objetos, como traslaciones, rotaciones o cambios de escala. Además, también posibilita el manejo de luces, colores o cámaras, de manera que se pueden controlar, a muy bajo nivel, todos los aspectos que se desee en cuanto a la representación del entorno virtual. Gracias a su fácil integración con distintos lenguajes de programación es posible dotar a los objetos del EV de comportamientos tan complejos como se pueda imaginar.

Es importante destacar que OpenGL sirve exclusivamente para la construcción de los entornos virtuales, no ofreciendo la posibilidad de crear y manejar ventanas, sonidos, conexión entre distintos clientes, etc., tareas que deberán ser realizadas mediante otros mecanismos que facilite el lenguaje de programación utilizado o mediante librerías adicionales.

Sobre OpenGL se han construido extensiones que proporcionan funciones de más alto nivel que permiten un uso más sencillo de esta librería sin necesidad de implicar al desarrollador en el manejo de detalles de bajo nivel que no le interesan. De especial interés resulta OpenGL Performer, una utilidad desarrollada específicamente para facilitar el desarrollo de simulaciones y aplicaciones de Realidad Virtual.

6.2.4 Desarrollo para web

Como casi cualquier otro tipo de aplicación actual, los EVs no han podido escapar del atractivo que ofrece Internet para su desarrollo y distribución, ya que la posibilidad de crear un EV que sea accesible a través de un navegador web hace que los potenciales usuarios del EV crezcan de manera exponencial. Para ello, basta añadir uno de los numerosos *plug-ins* existentes para los distintos navegadores que son capaces de visualizar el EV dentro de la ventana del navegador, como pueden ser:

- Cosmo Player: uno de los plug-ins más antiguos para visualización de VRML, aunque estuvo disponible para diversas plataformas, actualmente sólo se desarrolla para el sistema operativo IRIX, de Silicon Graphics.
- Cortona: ofrece soporte para VRML y Macromedia Flash, y está optimizado para su ejecución en PCs actuales. Se integra, además de con navegadores, con herramientas ofimáticas.
- Octaga: soporta VRML y X3D. En general, ha demostrado ser más rápido que el resto de plug-ins.
- blaxxun Contact: soporta VRML y X3D, y puede ser integrado fácilmente con otras aplicaciones.

Una de las mayores desventajas de estas aplicaciones se encuentra en su lentitud, pues una vez que se accede a Internet la velocidad de las comunicaciones es mucho menor que dentro de una red local. Cada vez que el usuario se conecta a un EV es necesario que éste se descargue en su máquina, lo cual suele llevar bastante tiempo. Si además el EV es multiusuario, la comunicación con el resto de usuarios se realizará con una velocidad bastante baja, motivo por el cual las aplicaciones que se realicen no podrán ser excesivamente dependientes de la velocidad de las comunicaciones.

VRML

VRML (*Virtual Reality Modeling Language*) es un lenguaje para describir simulaciones interactivas multiusuario. Su origen data del año 1995, cuando se publicó la primera especificación para VRML 1.0, cuyas capacidades de interacción quedaban bastante limitadas.

La especificación de VRML más actual es VRML97 [14], [15]. Aunque VRML está siendo substituido paulatinamente por X3D, su amplia utilización hace aún que sea interesante dedicarle alguna atención.

Como se define en el mencionado estándar, VRML es un formato de fichero para describir objetos y mundos tridimensionales interactivos. Por estar basado en la web, VRML permite que los objetos sean hiperenlaces a otros EVs, pero también a páginas web, imágenes, vídeos y otros tipos de documentos.

Los ficheros de VRML son archivos de texto que describen la geometría de los objetos 3D, su jerarquía y, si se desea, pueden incluir *scripts* que definan el comportamiento de los objetos. Sin embargo, estos comportamientos son de un tipo bastante elemental, y las posibilidades de interacción tampoco son demasiado sofisticadas, por lo que para sacarle todo el partido posible es necesario combinarlo con algún lenguaje de programación, generalmente Java.

La combinación de VRML y Java, ambos disponibles para una amplia variedad de plataformas, hace posible la construcción de EVs de una complejidad similar a la de otros construidos con otros lenguajes (v.g. C++ y OpenGL) con las mencionadas ventajas y desventajas de ejecutarse en un entorno web.

X3D

X3D (*eXtensible 3D*) es la evolución de VRML hacia un lenguaje de especificación de entornos tridimensionales [16].

El objetivo ha sido definir un nuevo lenguaje que cumpla una serie de requisitos que VRML no cumple, como separar la arquitectura de la codificación de los datos, añadir nuevos objetos y comportamientos o proporcionar distintas APIs para el manejo de los EVs.

En la actual especificación se contempla que X3D puede manejar gráficos 2D y 3D, animaciones, sonido y vídeo. También se da mayor soporte al manejo de dispositivos de interacción, control de la cámara dentro de la escena, mecanismos de detección de colisiones e integración con distintos lenguajes de programación (concretamente, ECMAScript y Java) y otros estándares como H-Anim o el protocolo DIS (*Distributed Interactive Simulation*).

Para mantener la compatibilidad con VRML se ha tomado la decisión de que los ficheros X3D se puedan definir de dos maneras. Una de ellas es respetando las sintaxis de VRML, mientras que la otra es a través del uso de una especificación en lenguaje XML.

Como se puede apreciar, por tanto, X3D continúa el camino emprendido por VRML, incorporando nuevos mecanismos para hacerlo más abierto, flexible y ampliable, pero manteniendo en esencia todas las características de VRML.

H-Anim

H-Anim es una especificación que originalmente surge para definir cómo construir figuras humanas con VRML [17]. Actualmente, con el desarrollo de X3D, se está definiendo un nuevo estándar que, entre otras características, presenta unas guías sobre cómo elaborar estas figuras con VRML y con X3D.

La necesidad de H-Anim surge del hecho de que existían diversas aplicaciones que servían para diseñar y animar figuras humanas. Aunque cada una realizaba su cometido con bastante éxito, la diferente representación de las figuras hacía bastante difícil la conversión de formatos entre distintas aplicaciones. Esto era especialmente notable al utilizar herramientas de modelado de un fabricante y aplicaciones de captura de movimientos de otro distinto. Con la adopción de H-Anim se facilita la tarea de integrar el resultado de ambos sistemas (v.g. para elaborar un avatar con movimientos similares a los humanos).

Algunos tipos de animaciones humanas no dependen de las dimensiones del cuerpo humano, mientras que otras sí. Teniendo esto en cuenta, es posible compartir las animaciones que no dependen de las dimensiones del cuerpo, mientras que si se mantienen las proporciones, también será posible adaptar las animaciones que sí dependen de las dimensiones del cuerpo humano.

Para tener todo esto en cuenta, H-Anim sugiere las dimensiones y las posiciones que deben tener todas las partes del cuerpo. Como además es habitual que en muchas aplicaciones no se requiera representar el cuerpo humano con el mismo grado de detalle, en el estándar H-Anim las articulaciones se han organizado en cuatro niveles de detalle, de manera que una animación hecha para una figura que se ajusta a un nivel de detalle será utilizable con figuras del mismo nivel de detalle o mayor.

En total, en la especificación actual de H-Anim se definen 76 características significativas en la anatomía del cuerpo humano, aunque tras su posterior traducción a articulaciones y objetos se pueden contabilizar hasta 89 elementos, por replicación de algunos puntos significativos.

6.2.4 Plataformas de desarrollo de EVs

Uno de los principales intereses en la construcción de EVs lo constituye la posibilidad de construir mundos que den soporte a múltiples usuarios de manera distribuida, y que estos mundos que se construyen sean fácilmente escalables.

A continuación se describen una serie de proyectos que han servido para dar un gran impulso al desarrollo de EVs con las mencionadas características.

Dive

DIVE (*Distributed Interactive Virtual Environment*) [8] es una herramienta que está orientada al desarrollo de EVs colaborativos de gran tamaño. Está especialmente pensada para soportar la comunicación de diversos clientes que se comunican a través de una red. Esta herramienta se ha desarrollado desde el año 1991, y actualmente se pueden encontrar versiones que se ejecutan sobre Windows, Linux, Solaris, Irix y HPUX.

DIVE es capaz de leer distintos formatos de objetos 3D, entre ellos VRML, y posteriormente el comportamiento de estos objetos se puede programar a través del uso de *scripts* escritos en Tcl. Estos *scripts* se pueden ejecutar como consecuencia de distintos eventos del sistema, como interacciones o colisiones, lo cual posibilita un comportamiento bastante complejo.

La comunicación está basada en un sistema P2P, de manera que no es necesario contar con un servidor de mundos, y toda la comunicación entre clientes se realiza mediante un proceso de multienvío y una base de datos distribuida del mundo. Esta base de datos se mantiene en cada cliente, de manera que la información del mundo perdura mientras al menos un cliente se encuentre activo.

Para que todo este volumen de comunicación funcione en tiempo real, y para que lo haga con redes cuyo ancho de banda no es muy grande, como sucede con Internet, se ha optado por la solución de no mantener una consistencia total entre las vistas del mundo en cada cliente, y se han implementado servicios que se aseguren de mantener un alto grado de consistencia, aunque ésta no sea total.

Existen varias opciones para utilizar DIVE en la construcción de un entorno virtual, entre las que se encuentran: codificar el EV en C/C++ o Java y compilarlo con las librerías de DIVE, usar DIVE como un *plug-in*, comunicarse con DIVE por TCP/IP o, como parecen preferir los autores, a través de *scripts* escritos en Tcl/Tk.

Massive

Massive es una serie de proyectos cuyo propósito es, en esencia, dar soporte al desarrollo de entornos virtuales multiusuario.

La primera versión de este proyecto [11] consistía en una aplicación de teleconferencia a la que se le dio la apariencia de un EV, y que era capaz de soportar alrededor de 10 usuarios, los cuales podían comunicarse con una mezcla de texto, sonido y el entorno 3D.

La segunda versión, Massive-2, también conocida como CVE (*CRG Virtual Environment*) [3], amplía y mejora las capacidades de la versión anterior, especialmente en lo que se refiere al modelo de interacción espacial. Para esta versión se desarrolló CRGShare, unas librerías escritas en C y diseñadas para ayudar a los desarrolladores que quieran implementar EVs distribuidos basados en Massive-2. Están estructuradas en forma de componentes, de manera que los desarrolladores puedan elegir los componentes que les convengan para su EV.

La actual versión, Massive-3, conocida como HIVEK (*HIVE Project Kernel*) [12], es un sistema de Realidad Virtual multiusuario y distribuido en el que se ha puesto un mayor énfasis en la escalabilidad. Al igual que sus predecesores, soporta comunicación con gráficos 3D y sonido en tiempo real, pero en este caso se ha eliminado el texto. Está desarrollado usando parte de las librerías que componen CRGShare, y gran parte de la arquitectura está basada en agentes.

Spline

Spline (*Scalable Platform for Large Interactive Networked Environments*) [34] es una herramienta cuya principal característica es ofrecer interfaces abiertas que faciliten la interoperabilidad entre EVs desarrollados de manera independiente.

Una de las características a destacar de esta herramienta es la definición del concepto de **locale** como forma de dividir un entorno en subentornos que se gestionan por separado, concepto que ha sido utilizado posteriormente en otros sistemas (v.g. Massive). Gracias a esta forma de construir los EVs, Spline hace posible que los usuarios puedan hacer cambios temporales o permanentes en el EV mientras éste se está ejecutando, de manera que puede evolucionar según los deseos de los usuarios.

La comunicación entre distintos usuarios y aplicaciones que hacen uso de un EV construido con Spline se hace a través de modificaciones en un modelo compartido del mundo virtual, donde se sabe la posición de cada objeto, su aspecto, qué acciones está realizando y qué sonidos está reproduciendo, y este modelo se reproduce en todos los clientes para mantener la eficiencia y la escalabilidad. El hecho de que el mundo esté dividido en *locales* hace que la información que hay replicar no sea excesivamente grande. Cuando en un cliente se realiza algún cambio, se comunica al resto a través de un proceso de multienvío.

Spline se ha utilizado para construir, como mundo más destacado, Diamond Park [34], un EV que representa una feria universal con edificios que recuerdan diversas culturas y periodos históricos y donde se pueden realizar actividades como diseñar paisajes o montar en bicicleta.

6.3 ¿Cómo abordar el desarrollo de un entorno virtual?

Hasta aquí se ha introducido al lector en el concepto de EV y los elementos que lo rodean; además se han revisado las principales herramientas que asisten al desarrollador en el proceso de implementación. El interés de este apartado es conciliar el desarrollo de este tipo de sistemas con la disciplina de la Ingeniería del Software, que tiene mucho que decir en cuanto a la manera óptima de plantear el desarrollo de estos sistemas para asegurar la calidad de los mismos.

En el caso de los desarrollos orientados a la construcción de EVs, el hecho de aplicar los procesos de desarrollo tal cual los proponen las diferentes metodologías existentes presenta problemas en forma de carencias en la manera de definir los requisitos, en la descripción visual de los EVs, y por lo tanto, en el diseño de éstos, problemas a la hora de gestionar los proyectos, sobre todo en la tarea de estimación, así como problemas de verificación del trabajo que debía realizarse, imposibilidad de reutilizar, etc. Con el fin de adaptar los modelos de proceso convencionales al desarrollo de un EV se describió el Marco Metodológico SENDA [29].

A lo largo de esta sección se describirá en primer lugar una visión global de procesos y su justificación dentro de SENDA; a continuación se describe en profundidad el proceso de Análisis, incluyendo ejemplos de aplicaciones en las que se ha llevado a cabo el proceso de Análisis propuesto en SENDA. Dado que este capítulo se centra en el proceso de análisis, si el lector desea conocer más detalles sobre el resto de tareas que propone SENDA para el resto de procesos de desarrollo puede consultar [30]. Las dos aplicaciones concretas sobre las que se basan los ejemplos que se incluyen a lo largo de esta sección son PRVIR, aplicación para el entrenamiento en centrales nucleares; y Escondite Inglés, para el entretenimiento a través de Internet.

6.3.1 Visión general del proceso

Debido a las características especiales de estos desarrollos, los procesos de Diseño e Implementación, se van a ver descompuestos como se indica en la Figura 6.3. El proceso de Diseño se descompone en los siguientes procesos: **Diseño 3D del EV**, **Diseño de Elementos Multimedia**, **Diseño de la Arquitectura Interna de los Componentes** y **Diseño del Sistema**. El proceso de Implementación se divide en dos procesos: **Implementación de Componentes de Soporte** e **Implementación del Módulo Principal**. El motivo de esta división en procesos se debe al tipo de tareas que en cada uno de ellos se realiza y a la relación existente entre procesos del modelo global. Gracias a esta descomposición se consigue mejor seguimiento y visibilidad sobre los desarrollos. No en todos los procesos son nuevas todas las tareas ni todas las técnicas propuestas; por este motivo se ha utilizado una notación para diferenciar los procesos en los que todas las tareas y técnicas propuestas son nuevas, los procesos en los que sólo algunas de las tareas o técnicas lo son, y aquellos en los que las tareas implican el uso de técnicas previamente descritas en otras disciplinas.

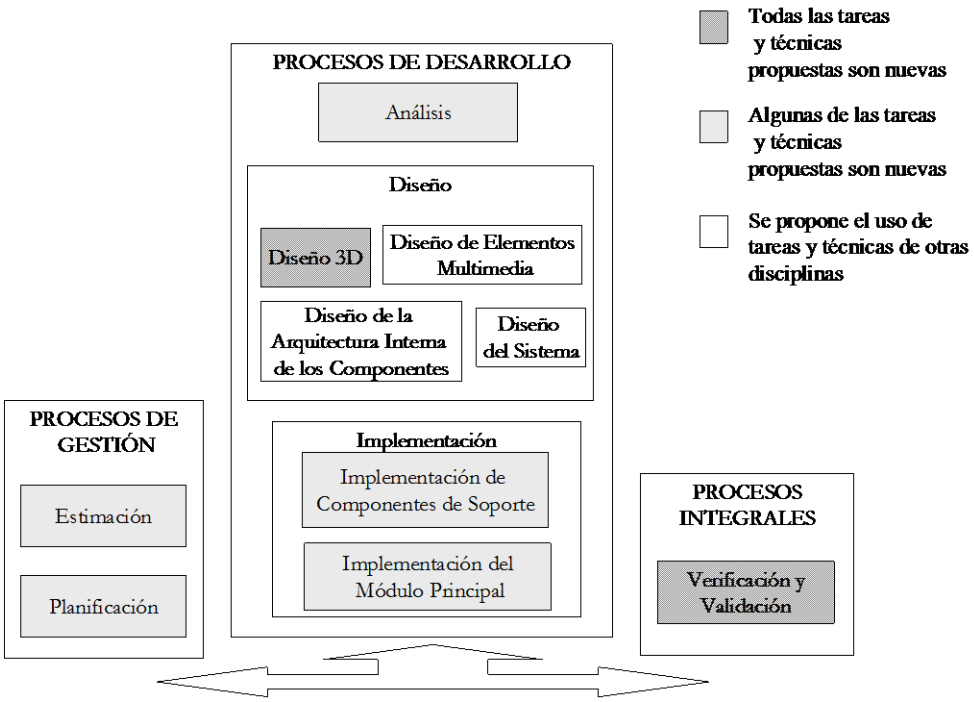


Figura 6.3. Modelo de Procesos Propuesto Detallado

Para la representación gráfica de las tareas y sus relaciones dentro de los procesos y con tareas de otros procesos se utilizará la notación de símbolos descrita en [20] para modelado de procesos. Para nombrar las tareas se utilizará una notación significativa compuesta por Acrónimo del Proceso + Acrónimo de la Tarea. Los acrónimos para los procesos son los siguientes:

- A: Proceso de Análisis
- D3D: Proceso de Diseño 3D
- DEM: Proceso de Diseño de Elementos Multimedia
- DS: Proceso de Diseño del Sistema
- DAI: Proceso de Diseño de la Arquitectura Interna de los Componentes
- ICS: Proceso de Implementación de los Componentes de Soporte.
- IMP: Proceso de Implementación del Módulo Principal
- E: Proceso de Estimación del Proyecto.
- P: Proceso de Planificación del Proyecto.
- V&V: Proceso de Verificación y Validación.

En la Figura 6.4 se muestran los procesos de la Figura 6.3 en más detalle, indicando las tareas que componen cada proceso así como la relación entre tareas dentro de cada proceso. La relación entre tareas de procesos distintos se describe en las distintas secciones dedicadas a cada proceso.

En todo sistema software que se quiera desarrollar, el primer paso en el desarrollo debe ser la extracción de requisitos. Concretamente en los EVs, debido a la evolución constante que están sufriendo, es preciso tomar una serie de decisiones tecnológicas que van a marcar el resto del desarrollo. Lo antes posible se debe decidir el tipo de dispositivos de realidad virtual, el software de desarrollo, el hardware, etc., con el fin de comprobar la compatibilidad de dichos elementos. Por esto se ha creado la tarea de **Definición de Requisitos Específicos**, dentro del proceso de Análisis. Así se puede reforzar la toma de dichas decisiones, que son específicas para EVs.

Además, como en todos los sistemas software, se debe hacer la extracción de requisitos funcionales. Una vez que se tenga la lista de requisitos del sistema, y tomando como axioma que la orientación a objetos es la que mejor se adapta a los EVs, se deben elaborar los casos de uso para todos aquellos requisitos que impliquen una interacción del usuario con el sistema, tal como se propone en [18]. Pero los casos de uso se quedan cortos al intentar representar determinados requisitos asociados a funcionalidad que no será demandada directamente por el usuario cuando haga uso del EV. Para resolver esta carencia, se propone una nueva técnica, los **conceptos de uso**, que resolverán el problema planteado por los servicios del EV que no son demandados directamente por el usuario.

Tanto los casos como los conceptos de uso se catalogarán basándonos en un conjunto de categorías que se proponen en el proceso de análisis. Dichas categorías vienen a agrupar las funcionalidades del sistema, basándose en los distintos modos de interacción identificados en la taxonomía de componentes propuesta en SENDA [31]. Así, se tendrán casos o conceptos de uso de percepción, animación, razonamiento, decisión, etc.

Además de los requisitos específicos y funcionales, existe otro conjunto de requisitos asociados al aspecto del EV. Dichos requisitos, como no son relativos a la funcionalidad del sistema, suelen quedar sin describir, al no tener cabida en el análisis de requisitos clásico. Para poder contemplarlos adecuadamente, se ha propuesto un proceso específico ubicado dentro de los de diseño, llamado **proceso de Diseño 3D**. Este proceso es necesario para poder alcanzar dos objetivos:

1. Extraer información sobre el aspecto que debe tener el EV
2. Poder comunicarse con el modelador, en un lenguaje común, que permita al diseñador del sistema y al modelador entenderse. En estos casos el lenguaje natural falla, debido a que la formación del diseñador de sistema (informático) y el modelador (experto en artes gráficas) es muy diferente.

Lo mismo ocurre con los elementos multimedia. Aunque son requisitos que tienen menos riesgo que los del modelado 3D del EV, también son requisitos de aspecto importantes. Por eso se ha creado el **proceso de Diseño de Elementos Multimedia.**

Una vez que se han extraído los requisitos de aspecto, se debe comprobar que dicha información especificada es la correcta. Para esto, se deben construir maquetas, vídeos, etc. Esto se ha incluido como parte del proceso de diseño del sistema, concretamente de la tarea de Diseño de la Interfaz. En dicha tarea se aprovechará para mostrar al usuario ejemplos de distintas alternativas de navegación por el EV. Esto es algo común a muchos sistemas software, pero es especialmente relevante en los EVs. Como bien se apuntaba en [5], la diferencia esencial entre la interfaz de un sistema tradicional y un EV es que, en el primero la interfaz sirve para ofrecer funcionalidad al usuario, mientras que en el segundo sirve además para conseguir que el usuario se sienta parte del EV. Las técnicas de diseño participativo o centradas en el usuario, son útiles para llevar a cabo esta tarea.

A partir del documento de conceptualización, en el que aparecerán todos los requisitos funcionales del EV, se extraerán clases para construir el modelo de estructura estática. Dicho modelo incluirá, además, clases provenientes del proceso de Diseño 3D y del Diseño de Elementos Multimedia. Tanto las clases como los métodos se obtienen de cada caso y concepto de uso, del proceso de Diseño 3D y del proceso de Diseño de Elementos Multimedia. Como los casos y conceptos de uso se han catalogado en categorías, será fácil saber de qué tipo de categoría es cada método. Es decir, si un caso de uso está catalogado dentro de la categoría de percepción, entonces deberá estar en alguna clase como método de detección, ya que es preciso dotar a la clase del correspondiente método de detección de modo que algo pueda ser percibido.

Dentro de las categorías de Casos y Conceptos de Uso, son especialmente críticas las que se refieren a lo que los componentes del EV pueden percibir, las que representan características internas, como personalidad, humor, etc., las que se refieren al modo en que razonarán y las que representan las reacciones posibles ante una interacción. Dichas reacciones pueden suponer desde una simple modificación en una variable hasta la representación de una escena muy compleja en el EV.

Para poder diseñar en detalle todo lo anterior, se propone un proceso de **Diseño de la Arquitectura Interna de los Componentes**. En dicho proceso se diseñará:

1. Cómo deben detectar o percibir los componentes del EV.
2. Cómo afectarán las características internas de los componentes al comportamiento de estos.
3. Cómo funcionarán los mecanismos de razonamiento partiendo de la detección de una interacción.
4. Cómo se representarán externamente las acciones de los componentes del EV.

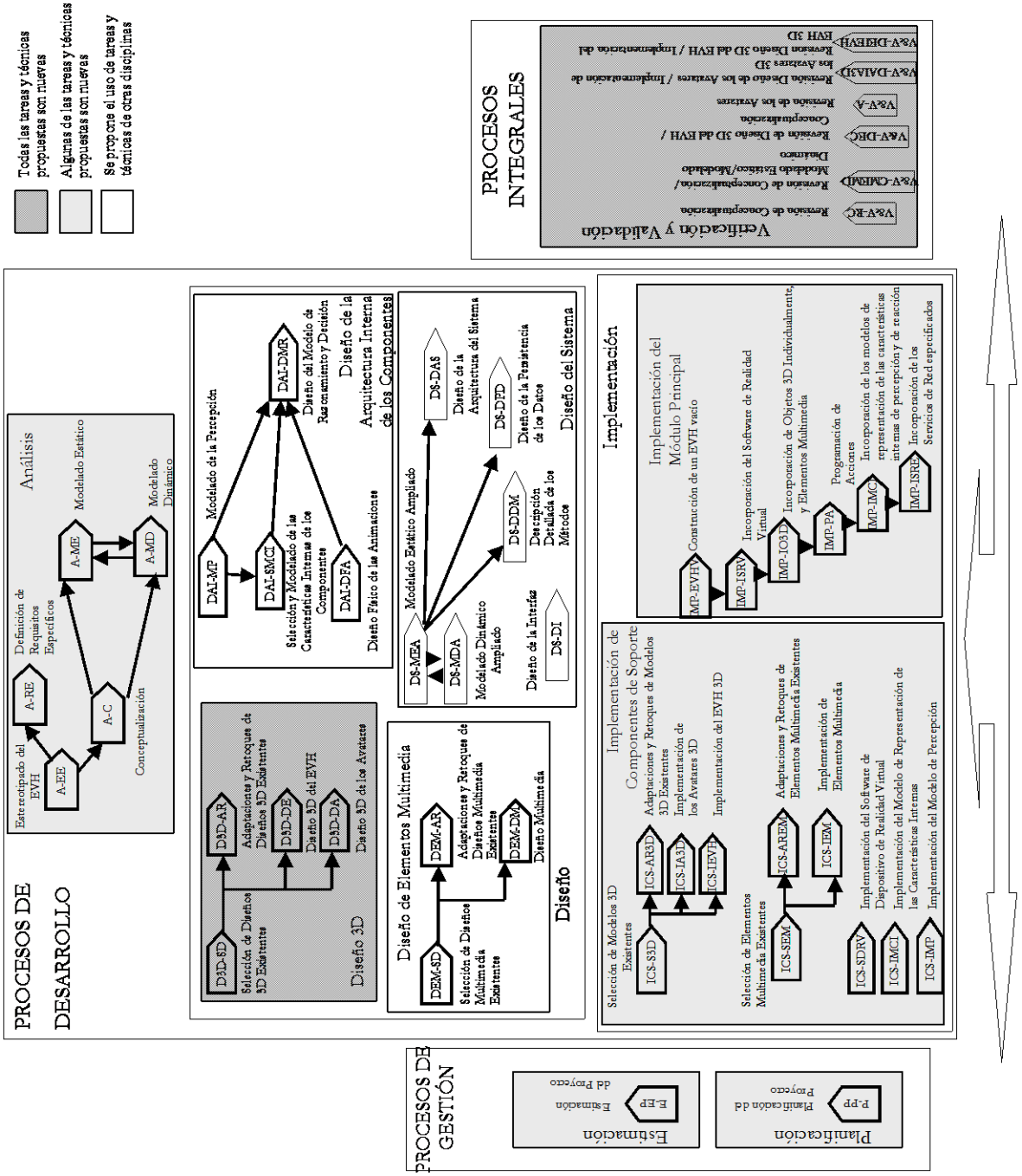


Figura 6.4. Conjunto de procesos y tareas del Marco Metodológico Propuesto

Dado que los EVs tienen muchos módulos o partes (v.g. el dispositivo de realidad virtual, los modelos 3D, elementos multimedia, etc.) que conforman todos ellos el EV, pero cuya implementación se puede llevar a cabo bien a partir de requisitos que se deciden muy al comienzo del proyecto o bien según van terminándose algunas tareas de diseño, y a su vez estos módulos no tienen relación entre sí hasta que no se integran en el EV definitivo, se ha propuesto un proceso de **Implementación de los Componentes de Soporte**, que incluye la implementación de todos estos módulos o partes de forma independiente.

A medida que dichos módulos vayan estando terminados, se irán integrando en el sistema, con el fin de ir mostrándole al cliente, lo antes posible, el EV definitivo. Para ello se ha creado un proceso de **Implementación del Módulo Principal**, que permite, de forma incremental, ir añadiendo pequeños módulos al EV. Por cada

porción del EV que se incorpore al sistema final, se le entregará una versión al usuario, de modo que si no es lo que esperaba se puedan hacer modificaciones a la parte que se acaba de incorporar.

A medida que se van realizando las tareas propuestas en SENDA es necesario ir verificando y validando los modelos que se van generando. Para ello se proponen un conjunto de tareas especiales de revisión incluidas en el proceso de Verificación y Validación.

6.3.2 El proceso de análisis en el desarrollo de EVs

Muchos investigadores sugieren que la fase de análisis debe contemplar una especificación de requisitos, o conceptualización, en la que se recojan exclusivamente características que definan qué hace el sistema, y nunca cómo debe comportarse el sistema [13]. Al igual que en [32], este trabajo comparte la perspectiva de que se trata de una idea muy atractiva, pero demasiado simplista en la práctica.

En los EVs, es especialmente relevante subrayar que se recogerán un conjunto de requisitos de muy diferente índole. Gracias a dichos requisitos se establecerán una gran variedad de características específicas del producto a desarrollar, concretamente estos requisitos permitirán determinar los valores de las características identificadas en la introducción de este capítulo y además el tipo de software que se utilizará como plataforma de desarrollo. De las decisiones que se tomen sobre dichos requisitos dependerán en gran medida otras fases del desarrollo (v.g. la estimación de coste y duración del proyecto dependerá de decisiones que se tomen en la fase de análisis).

Dentro del proceso de Análisis, se tendrán en consideración diferentes tareas que irán definiendo el sistema desde distintas aproximaciones. Como tarea inicial dentro del Proceso de Análisis se sugiere hacer un **Estereotipado del EV**, que servirá para perfilar adecuadamente las características del sistema. Para ello, se utilizarán unos cuestionarios de tipificación, cuyo contenido proporcionará al equipo desarrollador una visión más concreta sobre cuáles son las características específicas del proyecto a desarrollar, así como de las tareas que se deben realizar, ya que no todas las tareas del Marco Metodológico propuesto son necesarias para todos los proyectos.

El proceso de análisis no puede llevarse a cabo en su totalidad utilizando las técnicas tradicionales, ya que los casos de uso, utilizados en el análisis orientado a objetos, se centran sólo en las funcionalidades que el usuario puede demandar del sistema. Debido a las características de este tipo de sistemas interactivos, puede darse el caso de que algunas o muchas de las actividades que se desarrollan dentro de éste no sean demandadas por el propio usuario sino que sean automáticas. Para esto, es necesario definir los requisitos de algún modo fuera del alcance de las técnicas habituales basadas en análisis estructurado u orientado a objetos.

En el proceso de Análisis, concretamente en la tarea de conceptualización, se propone una forma adicional de definir los requisitos a través los llamados **Conceptos de Uso,** de manera que se contemple la posibilidad definir requisitos que no necesariamente impliquen la descripción de servicios que vayan a ser demandados por el usuario desde la interfaz.

Las tareas de Modelado Estático y Dinámico se han tomado de las metodologías orientadas a objetos, adaptándolas y relacionándolas adecuadamente con el resto de tareas de SENDA.

En la Tabla 6.1, aparecen las tareas de Análisis y en la Figura 6.5, la relación entre dichas tareas y los productos de tareas pertenecientes a otros procesos de SENDA.

Tabla 6.1. Tareas del proceso Análisis

	Tareas	Acrónimo
Proceso de Análisis	Estereotipado del EV	A-EE
	Definición de Requisitos Específicos	A-RE
	Conceptualización	A-C
	Modelado Estático	A-ME
	Modelado Dinámico	A-MD

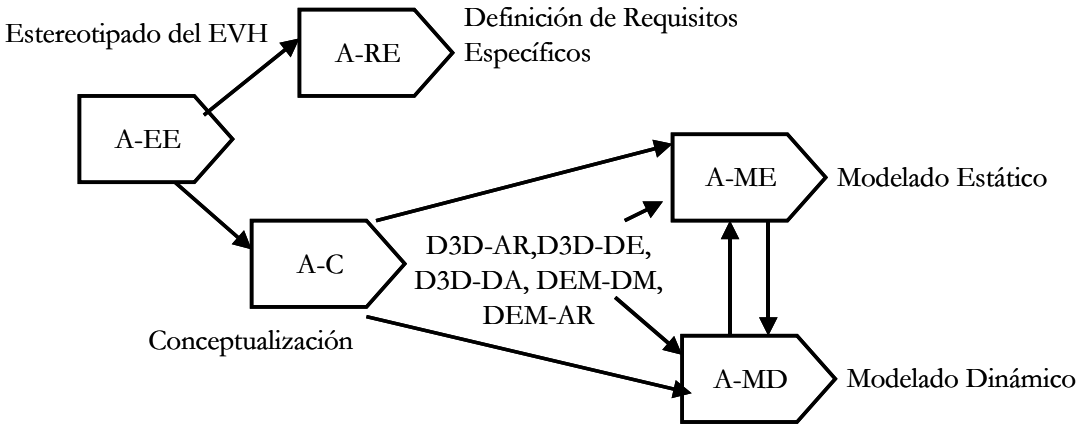


Figura 6.5. Relación entre tareas del proceso Análisis

A continuación se van a describir cada una de las tareas propuestas en el proceso de Análisis siguiendo una estructura común identificando productos de entrada de salida, así como técnicas y participantes para cada una de las tareas identificadas.

Estereotipado del EV

Se tratará de estereotipar el tipo de EV a construir a través de entrevistas con el cliente. Cada estereotipo de EV vendrá descrito por una serie de características y, como consecuencia de ello, habrá un conjunto de tareas asociadas, necesarias para llevar a cabo el desarrollo del EV de la forma más eficiente.

En la Tabla 6.2 aparecen los productos, técnicas y participantes correspondientes a esta tarea.

Tabla 6.2. Tabla de productos/técnicas/participantes de la tarea Estereotipado del EV

Productos	Entrada	Acuerdo con el cliente para iniciar el proyecto
	Salida	Estereotipo de EVa construir
		Mapa de Tareas
Técnicas		Entrevistas con el/los clientes
		Cuestionarios de tipificación
Participantes		Analista de Sistemas
		Clientes

La principal ventaja de esta tarea es la posibilidad que se le da al responsable del proyecto de construir un mapa de tareas propio para cada proyecto. Dicho mapa se puede construir a través de un conjunto de cuestiones que deben responderse sobre las características del EV que se quiere desarrollar. A partir de las respuestas del cuestionario se puede construir un mapa de tareas, a partir del cual se puede saber cuales de las tareas de SENDA han de llevarse a cabo y cuales no.

Esta tarea se propone con el fin de poder identificar, desde el principio del proyecto, el tipo de EV que se desarrollará. Además, permite al cliente asentar ideas sobre el producto que desea que le construyan, y al equipo de desarrollo tener más claras las tareas que deben llevar a cabo, y así se puede planificar mucho mejor el trabajo.

El cuestionario de tipificación que servirá para construir el mapa de tareas del EV aparece en la Tabla 6.3.

Una vez respondido el cuestionario, será responsabilidad de la persona encargada de gestionar el proyecto, construir el mapa de tareas que se llevarán a cabo. Para ello puede utilizar el mapa global que contiene todas las tareas de SENDA que parecen en la Figura 6.4.

Tabla 6.3. Cuestionario de tipificación del EV

¿El EV sólo servirá para realizar visitas guiadas, sin que exista ningún tipo de interacción?	**Sí** ❑ **No** ❑ Si la Respuesta es Sí, elimine del proceso de desarrollo el proceso de Diseño de las Características Internas de los Componentes del EV, y las tareas ICS-IMCI e ICS-IMP del proceso de Implementación de Componentes de Soporte y la tarea IMP-IMCI, del proceso de Implementación del Módulo Principal.

¿El EV funcionará en Red?	**Sí** ☐ **No** ☐ Si la respuesta es No, elimine la tarea IMP-ISRE del proceso de Implementación del Módulo Principal.
¿El EV utilizará dispositivos de realidad virtual?	**Sí** ☐ **No** ☐ Si la respuesta es No, elimine la tarea ICS-SDRV del proceso de Implementación de Componentes de Soporte y la tarea IMP-ISRV del proceso de Implementación del Módulo Principal.
¿El EV servirá para el aprendizaje?	**Sí** ☐ **No** ☐ Si la respuesta es Sí, deberá plantearse añadir a la arquitectura general de EV, un módulo tutor.
¿EL EV servirá para llevar a cabo relaciones sociales?	**Sí** ☐ **No** ☐ Si la respuesta es No, puede eliminar del proceso del proceso de Diseño de las Características Internas de los Componentes del EV, la tarea DAI-SMCI. Si la respuesta es Sí, deberá plantearse si necesita un modelo de personalidad o un modelo social para incluirlo en el EV.
¿El EV tendrá elementos 3D?	**Sí** ☐ **No** ☐ Si la respuesta es No, puede eliminar el proceso de Diseño 3D, y las tareas ICS-S3D, ICS-AR3D, ICS-IA3D, ICS-IEV del proceso de Implementación de Componentes de Soporte. Y de la tarea IMP-IO3D, recuerde que no debe realizar la parte correspondiente a la carga de elementos 3D en el EV.
¿El EV tendrá elementos multimedia?	**Sí** ☐ **No** ☐ Si la respuesta es No, puede eliminar el proceso de Diseño de Elementos Multimedia, y las tareas ICS-SEM, ICS-AREM, ICS-IEM del proceso de Implementación de Componentes de Soporte. Y de la tarea IMP-IO3D, recuerde que no debe realizar la parte correspondiente a inserción de elementos multimedia.
¿El EV tendrá avatares guiados por agentes?	**Sí** ☐ **No** ☐ Si la respuesta es Sí, los avatares deben ser modelados de tal modo que puedan ser manejados por agentes, es decir deben poder ser controlados desde la interfaz y desde dentro del sistema de forma automática, luego debe utilizar el formalismo de los Conceptos de Uso para definir algunos de los requisitos en la tarea de Conceptualización del proceso de Análisis.
¿El EV controlará total o parcialmente el modelo de personalidad para el avatar?	**Sí** ☐ **No** ☐ Si la respuesta es No, puede eliminar la tarea DAI-SMCI del proceso de Diseño de la arquitectura Interna de los Componentes y ICS-IMCI del proceso de Implementación de Componentes de Soporte.
¿El EV controlará total o parcialmente el modelo de razonamiento para el avatar?	**Sí** ☐ **No** ☐ Si la respuesta es No, puede eliminar la tarea DAI-DMR del proceso de Diseño de la arquitectura Interna de los Componentes.
¿El EV controlará total o parcialmente el modelo de percepción para el avatar?	**Sí** ☐ **No** ☐ Si la respuesta es No, puede eliminar la tarea DAI-MP del proceso de Diseño de la arquitectura Interna de los Componentes y ICS-IMP del proceso de Implementación de Componentes de Soporte.

Tarea de conceptualización
En la Tabla 6.4 aparecen los productos, técnicas y participantes correspondientes a esta tarea.

Tabla 6.4. Tabla de productos/técnicas/participantes de la tarea Conceptualización

Productos	Entrada	Estereotipo de EV
	Salida	Definición del Problema
		Definiciones, Acrónimos y Abreviaturas
		Lista inicial de requisitos funcionales del sistema
		Documento de Conceptualización, con casos de uso y conceptos de uso clasificados.
Técnicas		Conceptos de Uso
		Casos de Uso
Participantes		Analista de Sistemas
		Cliente
		Usuarios

En esta tarea se tiene que definir el problema y delimitarlo. Es importante definir claramente la magnitud del producto a construir, el ámbito en que se enmarca, el tipo de EV de que se trata, etc., teniendo en cuenta los resultados de la tarea anterior. Además, se debe hacer una lista de acrónimos y abreviaturas que se van a utilizar durante el desarrollo del sistema.

Con toda la información recopilada durante las entrevistas de la tarea de Estereotipado del EV, hay que hacer una lista en lenguaje natural que defina, en un párrafo, cada uno de los requisitos identificados, asignando para cada uno de ellos una referencia que lo distinga de los demás de forma unívoca.

Tras confeccionar la lista de requisitos anterior, se deben definir más precisamente los mismos. Para ello se utilizarán dos técnicas distintas. En el caso de los requisitos que provienen de una interacción entre el usuario y el sistema, es decir, aquellos en los que el usuario demanda una funcionalidad al sistema, se podrán refinar los requisitos utilizando para ello los Casos de Uso de Jacobson [18].

Sin embargo en los EVs se tiende a depositar cierto grado de control sobre el sistema, para que la interfaz con el usuario no esté repleta de controles que le dificulten la interacción. Será el cliente el que decida el grado de control que quiere que los usuarios tengan sobre el sistema, de manera que, si se decide delegar el control sobre el sistema, las funcionalidades ya no serán demandadas directamente por el usuario. Por este motivo los casos de uso, tal cual están definidos por [18], no sirven para especificar este tipo de requisitos. Para este tipo de requisitos este trabajo propone otro formalismo: los llamados **Conceptos de Uso**.

Un **Concepto de Uso** se redacta en una o dos frases y representa una de las posibles funcionalidades del sistema, no siendo estas funcionalidades demandadas directamente por el usuario sino delegadas en algún elemento del EV.

Determinar los conceptos de uso permite fijar rápidamente las funcionalidades que debe ofrecer el sistema al usuario. Cada concepto de uso suele coincidir con uno o varios requisitos previamente identificados. Puesto que los conceptos de uso están asociados a funcionalidades del sistema que no son demandadas directamente por el usuario, puede resultar útil describir la forma en que se deben realizar o la forma en que deben lanzarse dichas funcionalidades. Para ello, se propone asociar a cada concepto de uso un **Concepto de Operación**. Cada concepto de operación debe describir:

- El propósito: para qué se va a emplear dicho concepto de uso.
- El modo de funcionamiento: cómo se va a utilizar dicho concepto de uso.
- La frecuencia: frecuencia de utilización o frecuencia de ejecución de la funcionalidad.

Los conceptos de uso se van a representar del modo que aparece en la Tabla 6.5.

Tabla 6.5. Representación de un Concepto de Uso

CONCEPTO DE USO:	**CONCEPTO DE OPERACIÓN**
Nombre del concepto de uso: breve descripción. Código del concepto de uso: Concepto(XX), donde XX, será el número de orden en que se ha definido el concepto de uso.	Propósito: Modo de funcionamiento: Frecuencia:

De los tres atributos que definen un concepto de uso, sólo será preciso cumplimentar aquellos cuyos datos sean conocidos. Esto dependerá de los requisitos iniciales del proyecto. Los detalles de funcionamiento y frecuencia

son muy importantes para un tipo de funcionalidad identificada como Concepto de Uso, ya que será el sistema el que dispare esa funcionalidad de forma automática. Puede parecer extraño que se pida esta información en el análisis del sistema, pero en este trabajo se cree que si se conocen estos detalles, independientemente de si se está en la fase de análisis o diseño, deben salir a la luz lo antes posible y no ser guardados para un momento posterior.

En la Tabla 6.6 aparece un ejemplo de concepto de uso de la aplicación que implementa el juego del Escondite Inglés y en la Tabla 6.7 aparece un concepto de uso de la aplicación PRVIR.

Tabla 6.6. Concepto de Uso del EV del Escondite Ingles

CONCEPTO DE USO: Requisito 6	**CONCEPTO DE OPERACIÓN**
Nombre del concepto de uso: Un avatar detecta si está muy lejos o muy cerca de la pared. **Código del concepto de uso:** Concepto (2)	**Propósito:** Un determinado avatar deberá detectar su situación respecto de la pared, para así poder tomar una u otra determinación a la hora de moverse hacia ella (no correrá los mismos riesgos un avatar que esté cerca de la pared que uno que esté lejos) **Modo de funcionamiento:** En la pantalla se muestra la ubicación exacta de donde se encuentra la pared respecto del avatar. **Frecuencia:** Siempre que se mire a la pantalla deberá aparecer la situación de la pared, por lo que se tiene que actualizar constantemente.

Tabla 6.7. Concepto de Uso de la Aplicación PRVIR

CONCEPTO DE USO: Requisito 52	**CONCEPTO DE OPERACIÓN**
Nombre del concepto de uso: Atravesar torno a la entrada **Código del concepto de uso:** Concepto(14)	**Propósito:** Describir cómo debe atravesar al torno el avatar visitante cuando va a entrar a realizar un trabajo **Modo de funcionamiento:** Cuando el avatar visitante está frente al torno, para poder atravesarlo debe de introducir previamente el dosímetro y la tarjeta en la lectora, así como teclear el código de trabajo en la lectora. **Frecuencia:** Cada vez que un avatar visitante va a atravesar el torno a la entrada.

Es necesario clasificar los Conceptos de Uso y los Casos de Uso con el fin de poder, posteriormente, asignar adecuadamente métodos a clases. Para ello, se proporciona una clasificación inicial que está en función de las características generales de este tipo de sistemas. Por supuesto, sólo se utilizará lo que sea preciso de esta clasificación y, del mismo modo, pueden añadirse categorías en función de las necesidades del EV de que se trate.

En la Tabla 6.8 aparecen las diferentes categorías en que puede ser enmarcado un concepto o un caso de uso. Gracias a esta tabla el desarrollador puede agrupar las funcionalidades del sistema en tipos de interacción. Además, se podrá comprobar que están cubriéndose todas las características deseables para ese EV. También permite catalogar las funcionalidades de manera que posteriormente sea más fácil hacer su diseño e implementación (v.g. todas las funcionalidades que impliquen detección de algo están agrupadas, todas las acciones que deben representarse gráficamente también tiene su grupo correspondiente, etc.).

Tabla 6.8. Categorías generales propuestas. La presencia del simbolo * indica que no hay sub categorías identificadas

CATEGORÍA	**DESCRIPCIÓN**	**SUB CATEGORÍA:descripción**
C1	**De inicio de la conexión:** siempre y cuando se trate de un entorno multiusuario o cliente/servidor.	*
C2	**De interfaz con dispositivos de Realidad Virtual:** siempre y cuando se haya establecido como requisito el uso de dispositivos de Realidad Virtual.	*
C3	**De animación:** se asocian con los mecanismos de animación de los objetos reactivos y *proactivos&reactivos*.	**C3.1:** Movimiento o traslación de un elemento por el EV. **C3.2:** Expresión externa de alguna de las características

		internas de los elementos del EV. **C3.3:** De expresión de alguna acción (ya sea con el fin de interactuar o no con otro elemento del EV).
C4	**De percepción*:*** estos están relacionados con la capacidad de detectar lo que ocurre alrededor de un elemento del EV en caso de que éste sea reactivo o proactivo&reactivo. Los mecanismos de percepción o detección dentro del EV se podrían asemejar a la capacidad que tienen los humanos de percibir cosas en el mundo real.	**C4.1:** De detección de acción por parte de otro elemento del EV. **C4.2:** De detección de ubicación de otros elementos del EV. C4.3: **DE DETECCIÓN DE COLISIÓN**.
C5	**De evolución del EV:** esta categoría está relacionada con las necesidades de evolución del EV.	*
C6	**De razonamiento o decisión:** relacionadas con la actividad que se desarrolle en el EV. Seguramente, tras detectar algo, un elemento del EV debe razonar y tomar una decisión relacionada con aquello que ha detectado.	*
C7	**De comunicación con otros usuarios conectados:** en función del tipo de comunicación que van a poder establecer los usuarios a través de la aplicación. Por ejemplo, voz, *chat*, etc.	*
C8	**De Visualización de la escena**: si no existen mecanismos predefinidos para visualizar el EV, habrá que especificarlos y desarrollarlos.	*

Las categorías anteriores son generales para todos los sistemas. Una vez que se han identificado las categorías deseables para un sistema concreto, se debe rellenar una tabla cuyo formato aparece en la Tabla 6.9. Dicha tabla tiene cuatro columnas cuyo contenido se pasa a describir.

- Categoría: en esta columna aparecen todas las categorías seleccionadas para clasificar casos y conceptos de uso.
- Caso/concepto de uso que lo contempla: por cada categoría aparecerán los casos y/o conceptos de uso asociados.
- Nombre de la funcionalidad asociada: aparece por cada caso y concepto de uso el código o los códigos de los requisitos que cubre dicho caso o concepto de uso.
- Nombre de la clase en el modelo de clases: clase a la que se le asociará cada una de las anteriores funcionalidades.

Tabla 6.9. Tabla de clasificación de conceptos y casos de uso

CATEGORÍA	CASO/CONCEPTO DE USO QUE LO CONTEMPLA	NOMBRE DE LA FUNCIONALIDAD ASOCIADA	NOMBRE DE LA CLASE EN EL MODELO DE CLASES
….	….	….	….

La última columna de la tabla no se rellenará, como es lógico, en la tarea de conceptualización, sino como paso previo a la construcción del modelo de clases en la tarea de Modelado Estático.

Definición de requisitos específicos

En la
Tabla **6.10**, aparecen los productos, técnicas y participantes correspondientes a esta tarea.

Tabla 6.10. Tabla de productos/técnicas/participantes de la tarea Definición de Requisitos Específicos

Productos	Entrada	Definición del Problema
	Salida	Documento de Requisitos Específicos
Técnicas		Estudio de alternativas

	Entrevistas
Participantes	Analista de Sistemas
	Clientes
	Usuarios

Las aplicaciones basadas en EVs, requieren que se establezcan ciertos requisitos específicos en la fase de análisis. Gracias a la descripción de los requisitos específicos, el equipo de desarrollo puede ir preparando el entorno de desarrollo, e incluso cada uno de los implicados en el desarrollo puede ir entrenándose con las herramienta que tendrá que manejar llegado el momento. Es decir, los diseñadores gráficos pueden empezar a familiarizarse con la herramienta de diseño gráfico que se seleccione, los programadores con la herramienta de desarrollo y librerías, etc.

Dado que el desarrollo de EVs implica la toma de decisiones importantes y específicas al principio del proyecto, se propone un formato de documento que recoja, en fases tempranas del desarrollo, un conjunto de requisitos específicos. En la Tabla 6.11 se describen los apartados de dicho documento.

Puede parecer que es muy precipitado tomar estas decisiones al principio del proyecto, pero debe ser así con el fin de optimizar el resto de procesos, tanto de gestión como de desarrollo, ya que puede haber decisiones de hardware o software que requieran una fuerte inversión que es necesario contemplar desde el principio.

Se recomienda tomar una muestra suficientemente representativa de usuarios, y distribuirles una serie de cuestionarios sobre diferentes alternativas de interfaz. El motivo radica en el hecho de que en los EVs tiene una especial relevancia la interfaz, por lo que es mejor saber las preferencias de los usuarios antes de decidirse por una interfaz u otra. De este modo, es posible saber, por ejemplo, la predisposición que van a tener hacia el uso de un cierto dispositivo de realidad virtual. Lo ideal sería que los usuarios pudieran probar distintos mecanismos para interactuar con el sistema, como guantes de datos, casco, gafas, etc., pero esto no suele ser muy viable dado el precio de estos dispositivos.

Tabla 6.11. Descripción del documento de requisitos específicos

APARTADO	SECCIÓN	EXPLICACIÓN
1. Características de los usuarios		Es preciso indicar en este punto si los usuarios tienen algún tipo de discapacidad o preferencia que obligue a un tipo de interfaz u otro. Además, debe indicarse el rango de edad en que se encuentran los usuarios a los que va dirigido el sistema.
2. Requisitos de interfaz	Interfaz con el usuario	Los EV pueden tener multitud de interfaces diferentes, y hacer uso o no de distintos dispositivos de realidad virtual. Buena parte del sistema se verá influido por la selección de dichos dispositivos, y a su vez, los dispositivos de realidad virtual deben venir determinados por las características de los usuarios a los que va dirigido el EV, el objetivo de éste, así como el tipo de interacción requerida. Por ejemplo, si el EV está pensado para entrenar en la realización de tareas muy precisas en las que se requiere la utilización de las manos, casi seguro que la interfaz exija el uso de guantes de datos. Se debe indicar en este punto cómo será la interfaz de usuario a utilizar. Si se van a utilizar dispositivos de realidad virtual y si éstos requieren definir las características del entorno real, éstas también se describirán en este apartado.
	Interfaz con otros sistemas	Dado que se trata de sistemas complejos, es muy probable que deban existir interfaces con otros sistemas, incluyéndose aquí las interfaces con el software de los dispositivos de realidad virtual (en caso de haberlos), que generalmente es necesario desarrollar en paralelo con el propio EV. Además, puede que existan otros sistemas externos como tutores inteligentes, bases de datos, redes, etc.
	Selección del dispositivo de realidad virtual	El/los dispositivos de realidad virtual a utilizar deben quedar seleccionados en este punto. Habrá que estudiar detenidamente el tipo de usuario y el tipo de interfaz, de modo que los dispositivos seleccionados cumplan con los anteriores requisitos, en caso de haberlos.

3. Requisitos no funcionales	Requisitos hardware	Generalmente, las características de los EV hacen que los requisitos de hardware sean más estrictos que en otros sistemas. El hardware seleccionado debe ser tal que cumpla con los siguientes requisitos: Capacidad suficiente para visualizar escenas tridimensionales con suficiente calidad. Tanto la calidad de los objetos 3D como la velocidad de visualización se deben incluir como requisitos. Capacidad de funcionar en red. Capacidad de funcionar en tiempo real.
	Requisitos software	Se indicarán requisitos tanto de sistema operativo como de software, incluyendo dentro de este software el de los dispositivos de realidad virtual, el de diseño 3D, el de red y el de desarrollo propiamente dicho.
	Selección de hardware y software para el desarrollo	En función de los dos apartados anteriores se seleccionará el hardware y el software de desarrollo.
	Atributos de calidad	Si existen atributos especiales de tiempo de respuesta, rendimiento, portabilidad, fiabilidad, etc., se incluirán en este apartado.
4. Otros requisitos		En función del resultado de la tarea de estereotipado del EV, realizada en el proceso de análisis, se pueden ir definiendo con el cliente algunos aspectos relacionados con las características internas de los componentes del EV. Por ejemplo, se pueden ir decidiendo el modelo de percepción que se utilizará, las características específicas que afectarán a la arquitectura interna de dichos componentes, como pueden ser rasgos de personalidad, el estado de ánimo, el carácter, las intenciones, etc. Dichos modelos, pueden ser desarrollados explícitamente para el proyecto o ser reutilizados de otros proyectos.

Tarea de modelado estático

Esta tarea trata de obtener una imagen estática de los componentes del sistema. Estos componentes se modelarán mediante un modelo conceptual, utilizando para ello cualquier metodología basada en el paradigma de orientación a objetos [4], [19], [21], [28]. En este punto conviene recordar que esta tarea y la de Modelado Dinámico, se complementan; es decir, pueden producirse resultados en una, que afecten a la otra y viceversa, por lo que deben llevarse a cabo en paralelo.

En la Tabla 6.12 aparecen los productos, técnicas y participantes correspondientes a esta tarea.

Tabla 6.12. Tabla de productos/técnicas/participantes de la tarea Modelado Estático

Productos	Entrada	Documento de Conceptualización
		Todas las salidas del proceso de Diseño 3D
		Todas las salidas del proceso de Diseño de Elementos Multimedia
	Salida	Modelo de clases de análisis
		Tabla de clasificación de casos y conceptos de uso, de la tarea de conceptualización, ampliada.
Técnicas		Diagramas de estructura estática
Participantes		Analista de sistemas

De cada uno de los conceptos de uso, y de los casos de uso expandidos, se extraerán las clases potenciales, así como sus atributos, y las relaciones entre las clases identificadas. Para ello basta seguir cualquiera de los métodos de construcción de modelos de clases indicados en la bibliografía dedicada a la orientación a objetos [4], [28], [21], [19].

No sólo la tarea de Conceptualización contribuye a la construcción del modelo de clases. A partir de los productos de salida de los procesos Diseño 3D y Diseño de Elementos Multimedia, se podrá extraer información que aportará nuevas clases al modelo de clases. Así, los modelos 3D de los elementos que aparecerán en el EV serán clases del modelo de clases, mientras que los elementos de tipo sonido, vídeo, etc., pueden ser atributos de dichas clases.

La diferencia esencial entre las clases obtenidas a través de las funcionalidades descritas en el documento de conceptualización, y las del diseño 3D y de elementos multimedia es la que aparece en la Tabla 6.13.

Tabla 6.13. Diferencia entre clases obtenidas de diversas fuentes

Clase extraída del documento de conceptualización	Clase extraída del proceso de Diseño 3D	Clase extraída del Diseño de Elementos Multimedia
Suelen corresponderse con elementos no visibles.	Suelen corresponderse con elementos visibles (tienen representación 3D).	Suelen corresponderse con clases visibles, si se trata de imagen o vídeo, y no visibles pero sonoras, si se trata de audio.

Se propone la siguiente manera de proceder para extraer las clases y sus métodos asociados. Se extraerán los sustantivos y los verbos, de la descripción de los Casos y de los Conceptos de Uso, después se asociará a cada clase potencial, es decir, a cada sustantivo seleccionado, la funcionalidad identificada por el/los Concepto/s de Uso o Caso/s de Uso en los que ha aparecido identificada.

En este momento del desarrollo se puede volver a la tabla que se construyó en la tarea de Conceptualización, que aparece en la Tabla 6.9, y rellenar la columna que indica el nombre de la clase a la cual atribuimos el/los Concepto/s o el/los Caso/s de Uso.

Como ayuda a la identificación de clases se puede decir que serán clases potenciales los siguientes elementos del EV:

- El propio EV.
- Avatares
- Cerebro del avatar: donde se podrían ubicar los mecanismos de razonamiento y decisión.
- El cuerpo del avatar
- Por cada característica interna de los componentes del EV que se componga de más de un elemento, se construirá una clase. Es decir, si una característica interna es el humor, y éste se compone de grado de alegría y grado de enfado, por depender de más de un subelemento construiremos la clase humor.
- Todos aquellos objetos gráficos que tengan comportamiento.
- Si existen dispositivos de realidad virtual se debe incluir una clase gestora de éstos.
- Para poder almacenar la historia pasada de un avatar en el EV, es necesario definir la clase Memoria.
- Los mecanismos de percepción conviene agruparlos en clases que pueden recibir el nombre de los sentidos, en función de lo que sean capaces de percibir.
- Punto de vista para poder cambiar la posición desde la que se ve el EV.

También se propone una lista de relaciones típicas que se suelen dar entre elementos de un EV:

- El avatar tiene cuerpo, para poder expresar gráficamente las acciones.
- El avatar tiene cerebro, siempre que pueda tomar decisiones.
- El avatar tiene memoria, en la cual podrá guardar su historia pasada en el EV.
- El usuario puede tener al mismo tiempo varios dispositivo de realidad virtual seleccionados.
- El EV tiene elementos, que pueden ser de los tipos de la clasificación de componentes propuesta en este trabajo.
- El usuario tiene en cada momento un único punto de vista sobre el EV.
- El usuario puede usar varios dispositivos de conexión a la vez.

El modelo básico de clases que se propone es el que aparece en la Figura 6.6.

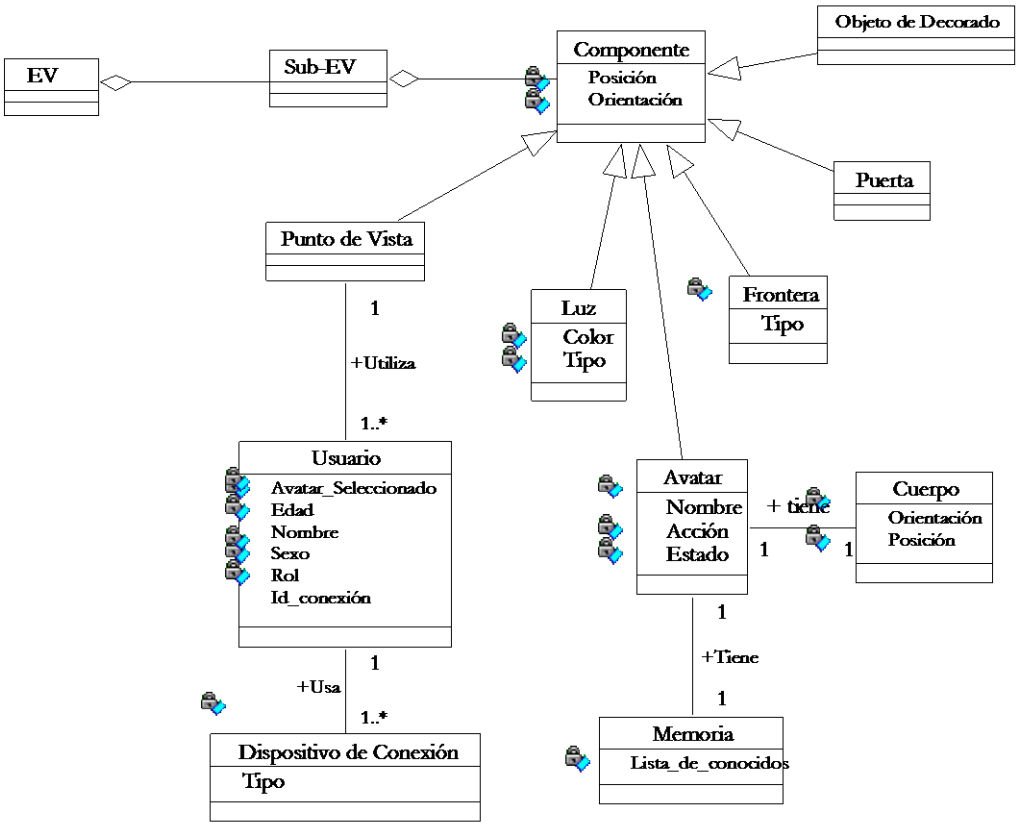


Figura 6.6. Modelo básico de clases

La Figura 6.7 representa el modelo de clases de la aplicación PRVIR.

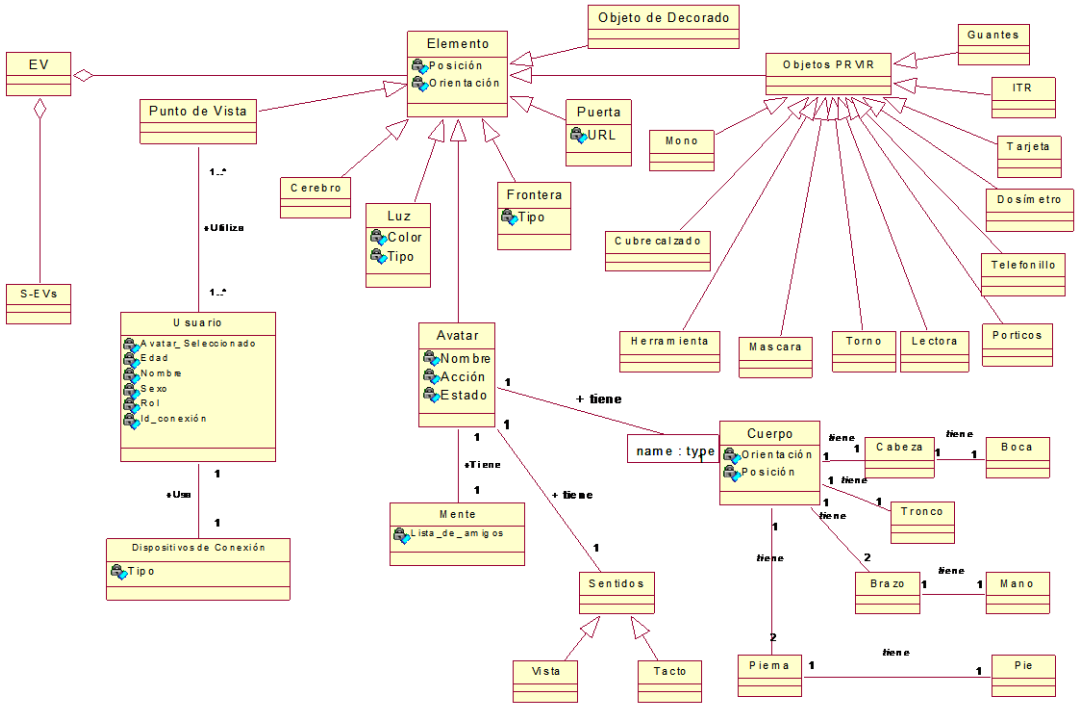


Figura 6.7. Modelo de clases de la aplicación PRVIR

Tarea de modelado dinámico

En este punto del desarrollo, la parte dinámica del sistema que puede analizarse varía sobre todo en función de la cantidad de información que se haya podido conseguir para construir los Conceptos de Uso. Cuanta más información se tenga, con mayor detalle se podrá describir la dinámica del sistema.

En la Tabla 6.14 aparecen los productos, técnicas y participantes correspondientes a esta tarea.

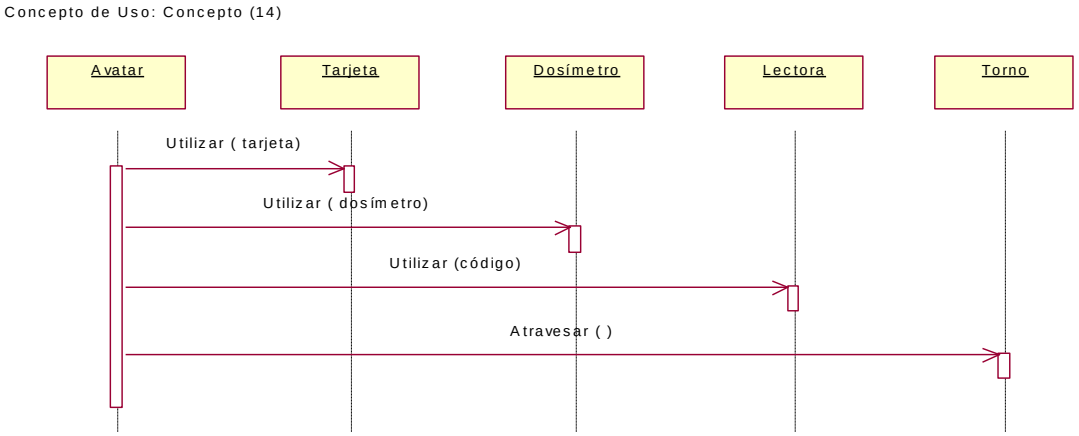
Tabla 6.14. Tabla de productos/técnicas/participantes de la tarea Modelado Dinámico

Productos	Entrada	Documento de Conceptualización
		Modelo de Clases de Análisis
	Salida	Modelo Dinámico
Técnicas		Diagramas de Secuencia del Sistema
		Escenarios
Participantes		Analista de Sistemas

Para los Casos de Uso obtenidos en el análisis, el tratamiento puede ser, por ejemplo, el propuesto en [21]: a partir de ellos se construyen los Diagramas de Secuencia del Sistema y los Contratos de Operación para describir el comportamiento del sistema.

En cambio, en el caso de los Conceptos de Uso no se puede utilizar directamente el método de Larman, ya que no los contempla. Los Diagramas de Secuencia del Sistema deben ser modificados de manera que sea una clase del sistema la que inicia los comportamientos, en lugar de un actor externo.

En la Figura 6.8 se incluye la descripción de la dinámica del Concepto de Uso de la aplicación PRVIR que aparece en la Tabla 6.7 en el que el Avatar se disponía a atravesar el torno a la entrada de la central nuclear.



DOCUMENTACION	
Avatar	Se corresponde con la representación del usuario dentro del entorno virtual.
Tarjeta	Es una tarjeta que el avatar visitante posee para su identificación.
Dosímetro	Se corresponde con el dosímetro empleado por un avatar visitante en sus tareas dentro de la zona radiológica
Lectora	Se corresponde con la lectora situada junto al torno para verificar el acceso del avatar visitante a la zona radiológica.
Torno	Se corresponde con el torno que da acceso a la zona radiológica.
Acciones	Utilizar(tarjeta): El avatar utiliza la tarjeta para que sea leída por la lectora.
	Utilizar(dosímetro): El avatar utiliza el dosímetro para que sea leído por la lectora.
	Utilizar(código): El avatar teclea su código de trabajo a realizar.
	Atravesar( ): El torno gira para que pueda ser atravesado.

Figura 6.8. Descripción dinámica del Concepto de Uso de la Tabla 6.7

Para las clases que representan las Características Internas de los Componentes del EV, es conveniente construir un Diagrama de Transición de Estados, ya que este tipo de diagramas es especialmente útil para clases cuyo comportamiento no sea trivial, como ocurre con las clases que representan características internas de los componentes del sistema.

6.4 Posibilidades de las aplicaciones de los Entornos Virtuales

Los entornos virtuales se han venido aplicando en numerosos dominios, de los cuales mencionaremos algunos ejemplos destacados. Las principales limitaciones para llevar estas aplicaciones a la web vienen dadas, hoy en día, por las restricciones de ancho de banda en la comunicación por la red, y por las limitaciones para el uso de dispositivos de realidad virtual inmersivos y de realidad aumentada. El hecho de funcionar sobre Internet puede plantear sobre todo problemas de velocidad, de seguridad y de fiabilidad. Estos problemas de velocidad, a su vez, pueden obligar a reducir el realismo y la interactividad del sistema. No obstante, son numerosos los ejemplos de entornos virtuales que se han desarrollado y utilizado con éxito para la web, en diferentes dominios de aplicación. Todo depende de los requisitos de cada sistema. En general cabe afirmar que cuanto mayores sean los requisitos de realismo de los modelos tridimensionales del entorno, y cuanto más compleja sean las posibilidades de interacción que el entorno debe ofrecer a sus usuarios, más difícil será conseguir un buen resultado a través de web. Así, las aplicaciones en medicina y en ingeniería suelen ser las más críticas.

6.4.1 Aprendizaje

Esta es quizá la aplicación más frecuente en la actualidad de los entornos virtuales y la realidad virtual. [23] [26].

Como entornos tridimensionales, se han aplicado con más éxito a situaciones de entrenamiento que a situaciones de formación. Estos entornos permiten a los estudiantes navegar e interactuar con una representación virtual de algún entorno real en el cual deben aprender a realizar alguna tarea. Son especialmente útiles cuando el entorno real no está disponible para el entrenamiento, o bien es muy costoso o arriesgado (v.g. una central nuclear [36]). Si el entorno virtual es multiusuario, permitirá además el entrenamiento de equipos de trabajo [1].

Un caso especial de entorno virtual para la formación podemos encontrarlo en algunas de las actuales plataformas de *e-learning*, que ofrecen a los estudiantes la posibilidad de reunirse virtualmente con otros estudiantes para discutir sobre las materias en estudio, o bien que ofrecen entornos de clases virtuales en las que el profesor imparte una clase de forma remota. Sin embargo, los estándares actuales de *e-learning* no están preparados para la inclusión, dentro del material formativo, de entornos virtuales tridimensionales, si no es como un objeto de aprendizaje completo e indivisible.

6.4.2 Trabajo colaborativo

Una de las aplicaciones más extendidas de los entornos virtuales, especialmente de los entornos multiusuario, es el trabajo colaborativo (*CSCW – Computer Supported Collaborative Work*). [2] [27]

Estos entornos ofrecen a los usuarios herramientas para compartir información, comunicarse y colaborar en la realización de determinadas tareas. Mediante la representación virtual de los demás usuarios, se facilita la toma de conciencia de la actividad que realizan los distintos miembros del equipo.

6.4.3 Medicina

La medicina es uno de los dominios en los que las aplicaciones de los entornos virtuales resultan de utilidad indiscutible, tanto para la práctica diaria de la medicina como para la formación de los futuros médicos [9] [35].

Así podemos encontrar aplicaciones de realidad aumentada en las que datos del paciente obtenidos mediante técnicas como la Tomografía Axial Computerizada o la Imagen por Resonancia Magnética son combinados con la vista del paciente real, dotando al médico de una especie de "visión de rayos X"; aplicaciones en las que los estudiantes de medicina intervienen a un paciente virtual con cirugía laparoscópica; etc.

6.4.4 Simulación, evaluación y predicción

Algunos EVs han sido desarrollados para simular determinados sistemas reales con el propósito de estudiar los mecanismos que regulan el funcionamiento de dichos sistemas, o bien para poder someterlos a determinadas

situaciones simuladas, estudiar la respuesta del sistema y evaluar su comportamiento, o bien para predecir cuál será el comportamiento del sistema real ante determinadas circunstancias.

Ejemplo destacado y especialmente interesante de estos sistemas son los que simulan multitudes humanas. Se han utilizado, por ejemplo, para simular la evacuación de una multitud de un gran edificio y evaluar diferentes tipos de sistemas de seguridad, caminos de evacuación, etc. [33].

6.4.5 Entretenimiento

Las posibilidades de los EVs en el mundo del entretenimiento son casi infinitas. Podemos encontrar entornos virtuales multiusuario en los que los visitantes pueden participar en actividades lúdicas tan simples como pasear y conversar con otros visitantes [34], o tan complejas como asistir a representaciones teatrales o de danza virtual [24].

Podemos encontrar aplicaciones de realidad aumentada donde un personaje virtual juega con nosotros al ajedrez o a un juego de cartas, o podemos hacer visitas turísticas virtuales a lugares lejanos, a reconstrucciones virtuales de lugares ya desaparecidos, o a sitios fantásticos donde podemos vivir todo tipo de aventuras.

6.4.6 Aplicaciones de los agentes virtuales

Un apartado independiente merecen las posibles aplicaciones de los agentes virtuales, que incluyen su participación en EVs culturales actuando como guías turísticos, en EVs de comercio electrónico actuando como vendedores, en EVs de formación y entrenamiento actuando como profesores, en EVs de entretenimiento como compañeros de juego, etc.

Esta es seguramente una de las áreas de los entornos virtuales más abiertas a la investigación, ya que en la actualidad aún estamos lejos de poder disponer de agentes virtuales realmente flexibles, inteligentes, y con potentes capacidades de interacción con los humanos [10]. En este sentido cabe destacar los trabajos sobre agentes virtuales conversacionales capaces de entablar diálogos con seres humanos [6], y las investigaciones sobre agentes pedagógicos [26].

6.5 Conclusiones

A lo largo de este capítulo se han descrito conceptos básicos relacionados con los EVs, de modo que tanto los noveles como los veteranos en esta área puedan empaparse de una terminología común utilizada a lo largo del resto del capítulo.

Dado el carácter multidisciplinar de este tipo de aplicaciones, de la carga de conceptos que puede albergar; sociales, psicológicos, etc., así como de las posibilidades interactivas que presentan, han sido durante los primeros años de su aparición objeto de una evolución tecnológica muy rápida. Esto es normal siempre que surge un tipo de aplicaciones novedosas y que brinden tantas posibilidades. Es por esto por lo que se construyeron multitud de plataformas de desarrollo, hablando en términos de implementación, algunas de las cuales hemos presentado al lector.

Una vez resueltas las cuestiones puramente técnicas, la disciplina de la Ingeniería del Software se presta como el vehículo idóneo para aunar experiencias en diferentes áreas: Diseño Gráfico, Interacción Persona Ordenador, Psicología, Sociología, etc., y definir de forma ordenada, rigurosa y repetible el modo y las técnicas que pueden asegurar el éxito en la construcción de un EV. El marco metodológico SENDA abarca el desarrollo completo de EVs, independientemente de la tecnología adoptada en cada caso. En este capítulo hemos querido resaltar el proceso de Análisis descrito en SENDA, proporcionando además ejemplos de aplicación del proceso de Análisis en proyectos reales.

Por último, tras exponer nuestra experiencia en este tipo de desarrollos, hacemos un recorrido por las principales áreas en las que los EVs están teniendo aplicación. Si bien esta breve revisión permite augurar un gran éxito, es cierto que aún nos enfrentamos a problemas a la hora de desarrollar estos sistemas, debidos algunos de éstos a cuestiones tecnológicas que, esperemos, poco a poco se irán resolviendo, pero también debidos a carencias metodológicas que en cierta medida hemos intentado paliar con este trabajo. También resaltar el hecho de que diseñar EVs para Web o sin Web es exactamente lo mismo pero actualmente existen problemas tecnológicos y de compatibilidad que dificultan el uso de estas tecnologías en la Web. Tal vez la relación de Web con televisión digital podría ser su punto de fusión que terminaría con muchos de los problemas tecnológicos actuales de desarrollo de EVs para Web.

Referencias

1. de Antonio, A., Ramírez, J., Méndez, G. (2004). An Agent-Based Architecture for Virtual Environments for Training. En Sánchez-Segura, M. (ed.) Developing Future Interactive Systems. Idea Group Publishers.
2. Benford, S., Fahlén, L. E. and Bowers, J. M. (1994) Managing Mutual Awareness in Collaborative Virtual Environments, en Singh, G.,Feiner, S. K., Thalmann, D. (eds.) Proceedings ACM SIGCHI Symposium on Virtual Reality Software and Technology (VRST'94), pp.223-236, World Scientific: Singapore
3. Benford, S., Greenhalgh, C., Lloyd, D. (1997) Crowded collaborative virtual environments. Proceedings of the SIGCHI Conference on Human factors in computing systems, p.59-66, March 22-27, 1997, Atlanta, Georgia, United States.
4. Booch, G. (1993). Object-oriented analysis and design with applications. Addyson-Wesley.
5. Bricken, M. (1990). Virtual Worlds: no Interface to Design. Human Interface Technology Center. University of Washington. Technical Report R-90-2.
6. Cassell, J.; Pelachaud, C.; Badler, N.; Steedman, M.; Achorn, B.; Becket, T.; Douville, B.; Prevost, S.; and Stone, M. (1994). Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. Proceedings of ACM SIGGRAPH '94. pp. 413-420
7. Frécon, E., Stenius, M. (1998). DIVE: A scalable network architecture for distributed virtual environments. Distributed Systems Engineering Journal (DSEJ) Vol. 5 , pp. 91-100, Special Issue on Distributed Virtual Environments.
8. Frécon, E. (2004) DIVE: Communication Architecture and Programming Model, IEEE Communications Magazine, Vol. 42, No. 4, April 2004.
9. Fuch, H. y otros. Augmented Reality Visualization for Laparoscopic Surgery. 1st International Conference Medical Image Computing and Computer-Asisted Intervention (MICCAI 98), Springer-Verlag, Heidelberg, Germany 1998, pp. 934-943.
10. Gratch, J., Rickel, J. et al (2002) Creating Interactive Virtual Humans: some assembly required. IEEE Intelligent systems, july/august 2002, pp.2-11.
11. Greenhalgh, C. M., and Benford, S. D. MASSIVE: A Virtual Reality System for Tele-conferencing, ACM Transactions on Computer Human Interfaces (TOCHI), 2 (3): 239-261, ISSN 1073-0516, ACM Press, September 1995.
12. Greenhalgh, C., Purbrick, J., Snowdon, D. (2000) Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring. Proceedings of the Third International Conference on Collaborative Virtual Environments (CVE 2000), pp. 119-127, ACM ISBN 1-58113-303-0. September 10-12 2000, San Francisco, CA.
13. Hsia, P, Davis, A., Kung, DC. (1993). *Status report: requirements engineering*. IEEE Software. Vol 10. Nº 6. pp. 75-79.
14. ISO/IEC 14772-1:1997. Information technology -- Computer graphics and image processing -- The Virtual Reality Modeling Language -- Part 1: Functional specification and UTF-8 encoding
15. ISO/IEC FDIS 14772-2:2004. Information technology -- Computer graphics and image processing -- The Virtual Reality Modeling Language (VRML) -- Part 2: External authoring interface (EAI)
16. ISO/IEC 19775-1:2004. Information technology - Computer graphics and image processing - Extensible 3D (X3D) - Part 1: Architecture and base components
17. ISO/IEC FCD 19774. Information technology - Computer graphics and image processing - Humanoid Animation (H-Anim)
18. Jacobson, I., et al. (1992) Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley.
19. Jacobson, I., Booch, G., Rumbaugh, J. (1999). The Unified Software Development Process. Addison-Wesley.
20. Kruchten, P. (1999). The Rational Unified Process. An Introduction. Addison-Wesley Object Technology Series.
21. Larman, C. (1998). Applying UML and patterns: an introduction to object-oriented analysis and design. Prentice Hall.
22. Macedonia, M., Pratt, D., & Zyda, M. (1994). NPSNET: A network software architecture for large scale virtual environments. Presence: Teleoperators and Virtual Environments, 3(4):265-287. Cambridge MA: MIT Press
23. McKay, D.P., Matuskey, P., Testani, S., et al. (1998). An Architecture for training virtual worlds environments. Proceedings of the virtual worlds and simulation conference. VSIM'98. Society for computer simulation, San Diego, USA
24. Meador, W. S., Kurt, E. M., O'Neal, K. R. (2003) Virtual performance and collaboration with improvisational dance. Proceedings of the SIGGRAPH 2003 conference on Sketches & applications: in conjunction with the 30th annual conference on Computer graphics and interactive techniques, San Diego, California
25. Milgram, P., Takemura, H., Utsumi, A., Kishino, F. (1994) Augmented Reality: A Class Of Displays On The Reality-Virtuality Continuum. SPIE Vol. 2351, Telemanipulator and Telepresence Technologies
26. Rickel, J., Johnson, W.L. (1999) Animated agents for procedural training in virtual reality: Perception, cognition and motor control. Applied Artificial Intelligence 13, pp.343–382.
27. Rodden, T. (1996) Populating the Application: A Model of Awareness for Cooperative Applications. ACM Conference on Computer Supported Cooperative Work (CSCW'96), Boston, Massachusetts, USA, ACM Press, pp. 87-96.
28. Rumbaugh, J. (1991). Object-Oriented Modeling and Design. Prentice-Hall International.
29. Sánchez-Segura, M. (2001). Aproximación Metodológica al Desarrollo de Entornos Virtuales Ph. D. Thesis. Technical University of Madrid. Spain.
30. Sánchez-Segura, M., et al. (2004). Developing Future Interactive Systems. Idea Group Publishers.
31. Sánchez-Segura, M., de Antonio, A, Amescua, A., (2004). Interaction patterns for future interactive systems components, Interacting with Computers (Vol 16/2 pp 331-350)
32. Sommerville, I., Sawyer, P. (1997). Requirements Engineering: a good practice guide. Wiley and Sons.
33. Thompson, P.A., Marchant, E.W. (1995) A Computer Model for the Evacuation of Large Building Populations. Fire Safety Journal, n. 24, pp-131-148.
34. Waters R.C., Anderson, D. B., Barrus, J. W., Brogan, D. C., Casey, M. A., McKeown, S. G., Nitta, T., Sterns, I. B., Yerazunis, W. S. (1997). Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability. MERL TR 96-02. Presence: Teleoperators and Virtual Environments 6(4): 461-480. Cambridge MA: MIT Press

35. Weghorst, S. Augmented Reality and Parkinson's Disease. Communications of the ACM, 40(8)
36. Youngblut, C. (1998). Educational Uses of Virtual Reality Technology. IDA Document D-2128. Alexandria, VA: Institute for Defense Analyses

7 Modelado de la Seguridad en Sistemas de Información Web

Daniel Sanz, Paloma Díaz, Ignacio Aedo

Laboratorio DEI. Departamento de Informática
Universidad Carlos III de Madrid.
Avda. de la Universidad 30, 28911 Leganés (Madrid)
dsanz@inf.uc3m.es, pdp@inf.uc3m.es, aedo@ia.uc3m.es
http://dei.inf.uc3m.es

7.1 Introducción

La seguridad en un sistema de información es un aspecto crucial. Con la aparición de Internet, el empuje por parte de las empresas y de la comunidad investigadora hacia la apertura y la interoperabilidad entre sistemas es imparable, lo que hace aún más necesaria la adecuada protección de los recursos de información y de procesamiento, que constituyen un activo cada vez más importante para las organizaciones. De hecho, para muchas de ellas resulta ser un factor clave en su competitividad, incluso para algunas es su razón de ser, por lo que el balance entre la adecuada prestación de servicios a través de la web y la protección de dichos activos adquiere carácter estratégico.

En este contexto, el diseño de la seguridad no puede ser un asunto de implantación, como ocurre en muchas ocasiones. La seguridad es un elemento de primer nivel, que entra en juego desde la concepción inicial del sistema, y participa desde el principio en las decisiones de diseño, como lo hacen otros tipos de requisitos (v.g. funcionales, de interfaz, etc.). Para lograr esta integración, se requiere considerar este tipo de requisitos explícitamente durante el proceso de desarrollo, lo que da lugar a la inclusión de fases o actividades dedicadas a la seguridad, cuyo resultado se materializa en forma de productos de diseño que pueden ser cotejados con el resto de vistas del sistema. Como se indica en [1], esta integración solventa la dificultad de acometer el análisis y diseño de la seguridad. A su vez, para poder realizar esto, se requiere de una base formal sólida que recoja los conceptos relevantes en el dominio de la protección, que toma la forma de modelos abstractos.

En el nivel de abstracción más alto, que es donde este capítulo se centra, uno de los servicios de seguridad más útiles es el control de accesos, que permite regular el modo en que los usuarios interactúan con el sistema. El modelado del acceso debe estar integrado con el resto del proceso de desarrollo [2], de forma que, desde el principio del proyecto, los desarrolladores sepan cómo expresar las reglas de acceso y cómo relacionar los modelos de acceso con el resto de vistas del sistema, evitando así conflictos o inconsistencias entre requisitos. Es por ello que este capítulo se centra en

el control de acceso y en cómo integrar el modelado del acceso en el desarrollo de sistemas hipermedia y por ende web.

La sección 7.2 introduce el papel del control de acceso en el contexto de la seguridad de los sistemas de información, describiendo algunos conceptos fundamentales como las medidas y mecanismos de protección, y particularizando en los sistemas hipermedia debido a sus especiales características. La sección 7.3 introduce conceptos como las políticas y modelos de seguridad, poniendo especial énfasis en el control de acceso basado en roles. También se enumeran principios de diseño útiles de cara al diseño de la seguridad en lo concerniente al control de acceso. La sección 7.4 resume el trabajo relacionado con el control de acceso y la hipermedia, incluyendo trabajos relacionados con la web y patrones de diseño de la seguridad. La sección 7.5 describe MARAH, el modelo de control de acceso basado en roles para hipermedia asumido por el método de diseño Ariadne (descrito en el capítulo 2). La sección 7.6 ilustra el uso de dicho método, en lo relativo a la seguridad, para el desarrollo del sistema ARCE. La sección 7.7 enumera las conclusiones y líneas de trabajo futuras.

7.2 Seguridad en las tecnologías de la información

La seguridad en un sistema de información comprende múltiples facetas, ya que es preciso proteger todos los elementos susceptibles de sufrir algún tipo de ataque, es decir, el software, el hardware y los datos. De nada sirve protegerlos parcialmente o dejar alguno sin protección. En general, el objetivo es mantener las tres características primordiales de la información:

- **`Confidencialidad`**: garantiza que la información es revelada sólo a los usuarios autorizados, en tiempo y forma precisa.
- **`Integridad`**: asegura que la modificación de la información es realizada por los usuarios habilitados, en el tiempo y forma precisa.
- **`Disponibilidad`**: permite que la información esté accesible, en tiempo y forma adecuada, a aquellos usuarios autorizados.

Las **`medidas de seguridad`** tienen como objetivo reducir los riesgos asociados al sistema de información. En general, el riesgo puede medirse en términos de las posibles amenazas, tanto externas como las que se pudieran producir dentro del sistema, las vulnerabilidades del sistema y del valor de la información y recursos contenidos en el mismo. Si bien las amenazas son difíciles de evitar, salvo algunas de carácter natural (v.g. inundaciones), y el valor del sistema no es posible reducirlo, la mejor forma de evitar riesgos es reduciendo las vulnerabilidades del sistema. Es por ello que las medidas de seguridad suelen centrarse en la eliminación o reducción de las vulnerabilidades, que representan las diferentes posibilidades con las que las amenazas pueden materializarse.

Atendiendo a la forma de actuación, las medidas de seguridad pueden ser de prevención, que tratan de evitar que la amenaza se materialice; de detección, que tratan de avisar del ataque; de corrección, que disminuyen las consecuencias del ataque, y de recuperación, que tratan de restituir el sistema al estado previo al ataque. Según su naturaleza, existen medidas legales, que protegen la información mediante

acciones penales; administrativo/organizativas, que hacen uso de medidas de administración y organización; físicas, que protegen los sistemas y su entorno de agentes físicos; y técnicas, que se adoptan dentro de los sistemas de información, protegen principalmente la información.

Las medidas de seguridad más relacionadas con los objetivos de este capítulo son las de carácter preventivo, en particular de tipo técnico y administrativo/organizativo, debido a la naturaleza multiusuario de los sistemas hipermedia y a la necesidad de administrar los derechos de acceso. Es importante resaltar que el ámbito de las medidas de seguridad suele exceder de las competencias del sistema como tal. Por ejemplo, para garantizar la disponibilidad hay que llegar más allá de un buen diseño ya que se depende de otros factores externos como podrían ser la cobertura de comunicaciones o el suministro de energía eléctrica.

Entre las medidas de seguridad de carácter técnico se encuentran:

- **Identificación y autenticación de usuarios**: mientras que la identificación pretende obtener la identidad del usuario que accede al sistema, la autenticación pretende confirmar que el usuario es quien dice ser. Normalmente la autentificación se realiza, bien por algo que se tiene (v.g. una tarjeta), bien por algo que se sabe (contraseña) o bien por algo que se es (características de la persona, como la huella digital).
- **Control de accesos**: una vez confirmada la entrada del usuario en el sistema, el control de accesos pretende asegurar que las acciones realizadas por el usuario están en conformidad con los privilegios del mismo.
- **Control del flujo de la información**: complementa al control de accesos, evitando ciertos actos de los usuarios sobre los datos a los que tienen derecho a acceder. Por ejemplo, puede evitar la copia de un fichero de acceso restringido a uno sin restricciones de acceso.
- **Confidencialidad**: pretende evitar el acceso a la información por parte de usuarios no autorizados.
- **Integridad**: pretende evitar la modificación de la información por parte de usuarios no autorizados.
- **No repudio**: evita que un sujeto reniegue de la realización de una acción que previamente sí había efectuado.
- **Notorización**: ofrecen confiabilidad, mediante la certificación de la asociación entre individuos y claves públicas de cifrado.
- **Auditoría**: registran todas las acciones realizadas en el sistema por parte de lo usuarios.

Entre las medidas administrativo/organizativas se encuentran la clasificación de la información de acuerdo con el nivel de confidencialidad, asignación de responsabilidades a los usuarios, establecimiento de funciones de seguridad, formación y sensibilización de los usuarios, etc. Como se verá más adelante, el control de acceso requiere una adecuada elección de muchas de estas medidas.

Conceptualmente, hay que diferenciar las medidas de seguridad de los **`mecanismos`**

de protección. Si bien aquéllas son requisitos exigidos a un sistema para considerarlo seguro, es decir, forman parte de la **política de seguridad** definida para el sistema, éstos son realmente los encargados de asegurar el cumplimiento de las medidas, representando la puesta en práctica de una determinada política de seguridad. El concepto de política de seguridad será tratado con más detalle en la siguiente sección.

Los mecanismos de protección entran en juego dentro del ámbito de diseño del sistema, afectando a uno o varios componentes del mismo. Pretenden garantizar que los usuarios sólo acceden a sus propios recursos, y que los recursos sólo son accedidos por aquellos usuarios que tienen derecho de acceso a los mismos. Los principales mecanismos de protección son:

- **Autenticación**: proporciona identificación y autenticación de usuarios.
- **Control de accesos**: proporciona control de accesos y del flujo de la información.
- **Cifrado de datos**: proporciona confidencialidad.
- **Funciones resumen**: se encargan de garantizar la integridad de los datos.
- **Firma digital**: garantiza el no repudio.
- **Registro de auditoria**: proporciona medidas de auditoria.

En lo concerniente a los sistemas hipermedia, debido al hecho de que la mayoría de ellos se materializan como sistemas web, surgen ciertas implicaciones de cara a los mecanismos de protección:

- La adopción de la arquitectura cliente-servidor es masiva, lo que da lugar a mecanismos para proporcionar seguridad a nivel de comunicaciones en la red, relacionados con la confidencialidad principalmente (v.g. cifrado).
- El gran uso de estos sistemas para realizar transacciones comerciales electrónicas, como compras o movimientos de dinero, requiere proporcionar servicios de autenticación, integridad, no repudio, auditoria y confiabilidad.
- La diversificación de los usuarios da lugar a la necesidad de controlar el acceso a los recursos de manera que la información sea accedida y manipulada adecuadamente de acuerdo a las competencias, responsabilidades o necesidades de cada usuario. En este sentido, estos mecanismos, además de proporcionar protección, permiten adecuar o personalizar el sistema en función del usuario, lo cual es un aspecto importante en la hipermedia, en particular de cara a la usabilidad del sistema.

Claramente, estos aspectos no deben dejarse de lado hasta la implementación del sistema, sino que deben tenerse en cuenta mucho antes, si se desea construir sistemas fiables y usables. Es por estos motivos por los que se hace necesario una integración de los requisitos de seguridad con el resto del desarrollo del sistema hipermedia. En particular, el papel del control de acceso cobra especial relevancia por tener la doble función de protección y personalización, y por requerir, a diferencia de otros mecanismos como el cifrado o la firma digital, de modelos específicos para hipermedia que tengan en cuenta las características de estos sistemas.

7.3 Diseño de políticas de seguridad

La existencia de un sistema seguro pasa por la definición de una **política de seguridad** que defina claramente los aspectos de seguridad que el sistema proporciona, independientemente de los mecanismos utilizados para implementarla. Los requisitos de seguridad son una cuestión importante para las organizaciones, y normalmente varían de una organización a otra. También varían de un sistema a otro, o incluso dentro de un sistema, de unos usuarios a otros. Por eso, se dice que un sistema es **confiable** con respecto a una determinada política de seguridad si ofrece mecanismos de protección capaces de cumplir con los requisitos de seguridad impuestos por dicha política.

De cara al control de acceso es importante introducir el término **monitor de referencia** (*reference monitor*), que representa una parte abstracta del sistema, encargada de mediar en las peticiones de acceso al mismo, y tomando una decisión de control de acceso. La implementación software o hardware del monitor de referencia se denomina **núcleo de seguridad** (*security kernel*). Así, al conjunto de elementos hardware, software y otros, responsable de asegurar el cumplimiento de una política de seguridad determinada se le denomina **base de computación confiable** (TCB, *Trusted Computing Base*). La TCB incluye, entre otros mecanismos de protección, al monitor de referencia. Si bien estos términos, acuñados en el ámbito de los sistemas operativos, son perfectamente válidos, la protección en sistemas distribuidos requiere la existencia de diversos puntos de comprobación y cumplimiento. Por ello, se diferencia entre el **punto de decisión de la política** (PDP, *Policy Decision Point*), en el que se toma una decisión de control de acceso, y el **punto de cumplimiento de la política** (PEP, *Policy Enforcement Point*), en el que se efectúa el control de acceso como tal de acuerdo con la respuesta emitida por el PDP, pudiendo existir varios PDPs y PEPs repartidos por el sistema [3].

El ejemplo de política de seguridad más conocido es la política militar. Clasifica cada objeto del sistema (v.g. un fichero) en uno de los cinco niveles de seguridad clásicos: desclasificado, restringido, confidencial, secreto y alto secreto. Estos niveles se estructuran lógicamente como círculos concéntricos, en cuyo interior está el nivel de alto secreto. Los sujetos (v.g. usuarios) se clasifican en función del nivel al que pueden acceder, de forma que, en general, un sujeto con acceso a objetos del nivel `i` gozará también de accesos a objetos del nivel `i+1`. Además, se utiliza la regla de "lo que se necesita saber", es decir, sólo se permite el acceso a los datos sensibles a quien los necesita para su trabajo. Por ello, los sujetos se estructuran en compartimentos que restringen más la regla general de acceso, aplicándose ésta dentro de cada compartimento. Los compartimentos permiten ocultar un objeto a un sujeto con el mismo nivel de seguridad que el objeto. Así, los objetos se clasifican asignándoles un valor de la tupla <`nivel, compartimento`>. La relación de dominancia regula el acceso de los sujetos a los objetos mediante la comparación del nivel de confidencialidad y del compartimento del sujeto y del objeto al que pretende acceder.

De esta forma, se dice que un sujeto puede acceder a un objeto sólo si su dominancia es mayor.

Las siguientes secciones exploran conceptos relacionados con el diseño de políticas de seguridad, como son los modelos de seguridad, los principios de diseño, el control de acceso basado en roles y los patrones de diseño de la seguridad.

7.3.1 Modelos de seguridad

Un modelo de seguridad es un mecanismo abstracto que permite poner en práctica una determinada política de seguridad. Los modelos permiten probar la completitud y coherencia de la política, así como facilitar el diseño del sistema y comprobar si la implementación cumple con todos los requisitos de seguridad. Desde el punto de vista de la ingeniería del software, los modelos ofrecen una base formal para elaborar un diseño, y son claves para recoger los requisitos de seguridad de una forma completa y no ambigua.

En relación con el control de acceso, tradicionalmente se ha venido diferenciando entre los modelos de control de acceso obligatorio (MAC, *Mandatory Access Control*), y los de control de acceso discrecional (DAC, *Discretionary Access Control*) [4].

Los modelos MAC son modelos de seguridad multinivel, que representan la información estructurada en rangos de sensibilidad, nombrados mediante etiquetas. Los flujos de información permitidos se reflejan gráficamente en forma de rejilla (*lattice*), conectando las etiquetas o niveles entre cuyos objetos se permite el flujo de información. La rejilla es, pues, un orden parcial que define dichos flujos. Los sujetos se organizan jerárquicamente usando las mismas etiquetas, para restringir el flujo de información en función de sus acreditaciones (v.g. la política militar). Un ejemplo de MAC es el modelo de Bell-LaPadula [5].

Los modelos DAC, también llamados modelos de seguridad limitada, no se centran en prevenir el flujo de la información, sino que se basan en que el dueño de un objeto, que coincide habitualmente con su autor, tiene el control sobre los permisos de ese objeto, de ahí que los administre a su discreción. Los permisos se almacenan en una **`matriz de accesos`**, en cuyas filas se sitúan los sujetos, y en cuyas columnas están los objetos del sistema. Así, los derechos de acceso del sujeto `i` sobre el objeto `j` están almacenados en la celda (`i, j`) de la matriz. Cuando la matriz se descompone en columnas, para cada objeto se tienen todos los modos de acceso por cada sujeto, por lo que se tiene un modelo basado en autoridad. Si, en cambio, se descompone por filas, son los sujetos los que saben qué pueden hacer sobre cada objeto, por lo que se tiene un modelo basado en capacidades. Un ejemplo de DAC es el modelo de Harrison-Ruzzo-Ullman [6].

Una alternativa a estos modelos es el control de acceso basado en roles (RBAC, *Role-Based Access Control*), que utiliza el concepto de rol para agrupar un conjunto de

permisos y un conjunto de usuarios autorizados a ejercer dichos permisos. Debido a la gran relevancia de este modelo, y a que constituye la base formal de gran parte de los contenidos de este capítulo, sus detalles se explican en una sección posterior.

7.3.2 Principios de diseño

Como se comentó anteriormente, el concepto de seguridad total es inalcanzable. Sencillamente, no existe un sistema cien por cien seguro, por lo que el esfuerzo se centra en lograr sistemas confiables, en el sentido de garantizar los requisitos de seguridad de la organización y de generar confianza en los usuarios. La experiencia en el desarrollo de mecanismos de seguridad, en concreto relacionados con el control de acceso, ha dado lugar a los siguientes criterios de diseño [7,8]:

- **Abstracción de datos**: los mecanismos de protección deben definirse usando elementos del nivel de abstracción adecuado, evitando alejarse del dominio de aplicación. Así, al especificar permisos sobre una cuenta bancaria es preferible hablar de "ingresar" y "retirar" antes que de "leer" y "escribir" en un fichero de datos que almacene los movimientos.
- **Privilegios mínimos**: deberán asignarse a los usuarios los mínimos privilegios necesarios para acometer sus tareas, y ninguno más.
- **Separación de privilegios**: las tareas críticas del sistema deben diseñarse de forma que sean realizadas por más de una persona, dificultando la posibilidad de uso fraudulento del sistema.
- **Separación de administración y acceso**: la administración de la política de acceso debe estar separada del acceso a la información del sistema. Además, que el administrador pueda dar un permiso no le habilita para ejercer ese permiso.
- **Autorizaciones positivas y negativas**: para añadir flexibilidad, deberán asumirse autorizaciones tanto positivas, que permiten el acceso, como negativas, que deniegan el acceso.
- **Delegación de privilegios**: debe ser posible delegar tareas administrativas a los usuarios, cuando éstas no sean críticas para el funcionamiento del sistema, o si así lo determina la política de seguridad.
- **Transacciones bien formadas**: las operaciones que manipulan objetos son conocidas, su comportamiento es predecible y carecen de errores, por lo que tras su aplicación el estado del sistema permanecerá consistente. Además, sólo puede accederse al mismo a través de dichas operaciones.
- **Autenticación**: se asumen usuarios correctamente autenticados. De esta forma se separa el control de acceso del problema de determinar y verificar la identidad del usuario.
- **Compartición mínima**: debido a que los objetos compartidos pueden servir como canales de comunicación de información, debe aplicarse el principio de separación (física o lógica) siempre que sea posible. Así se dificultan los flujos de información no deseados.
- **Diseño abierto**: el diseño del sistema debe ser público, lo que permite ser estudiado por cualquiera, aportando verificaciones independientes del

sistema. Al reducir los aspectos ocultos, se deja de asumir la ignorancia de los intrusos.

- **Exigencia de permisos**: por defecto, el acceso debe ser restrictivo. Deben pedirse permisos de acceso para todos los objetos del sistema.
- **Intermediación completa**: debe comprobarse cada intento de acceso al sistema, de forma que los mecanismos de comprobación no puedan ser evitados.
- **Mecanismos económicos**: los mecanismos de protección deben ser sencillos, regulares y pequeños, para facilitar el análisis, diseño y la verificación.
- **Sencillez de uso y aceptabilidad**: los mecanismos de protección deben ser de fácil aplicación y aceptados por parte de los usuarios, así se reduce la posibilidad de que los usuarios traten de evitarlos.

7.3.3 Control de acceso basado en roles (RBAC)

El propósito de cualquier mecanismo de control de acceso es proteger los recursos de un sistema limitando el acceso a los mismos [9]. Por acceso se entiende un tipo de interacción que puede darse entre usuario y sistema y que da lugar a un flujo de información entre ambos. El control de acceso basado en roles es un tecnología nueva y a la vez vieja: si bien los modelos formales han venido apareciendo recientemente, ya entrados en los años 1990, el concepto y el uso del rol viene practicándose desde mucho antes como forma de gestionar los privilegios. Los roles pueden representar tareas, responsabilidades, obligaciones, capacidades o necesidades asociadas con el desempeño de una función en el trabajo.

El concepto de RBAC es simple, se trata de definir los permisos basándose en los roles establecidos en la organización, para luego asociar adecuadamente a los usuarios con los roles que tengan derecho a ejercer. De esta forma, la decisión de acceso está basada en los roles que los usuarios individuales tienen como parte de una organización.

Debido a que los roles son bastante más persistentes, en términos de permisos, que los usuarios, debido por ejemplo a rotación de personal o a reasignación de tareas, RBAC ofrece un poderoso mecanismo para reducir la complejidad, el coste y la probabilidad de cometer errores en la asignación de permisos a los usuarios. Esto se debe a que permite al administrador especificar reglas usando la estructura organizativa de forma natural, y a su vez, modificar los usuarios asignados a un rol, reflejando cambios en las asignaciones de trabajo de éstos, o, menos probable, modificar los permisos de un rol, reflejando cambios en funciones organizativas.

RBAC permite la especificación y el cumplimiento de una gran variedad de políticas de seguridad, que son el resultado de configurar adecuadamente los distintos componentes de RBAC. De hecho, se dice que RBAC es neutral con respecto a la política, ya que permite simular otros modelos previos [4], que han demostrado limitaciones, como son DAC o MAC.

Desde finales de los años 90 se viene reconociendo la necesidad de unificar conceptos y terminología en torno a RBAC, ya que no había una aceptación común del concepto RBAC, y sí una variedad de interpretaciones, usos e implementaciones por parte de distintos fabricantes. Por ello, se inició un esfuerzo por parte del NIST (*National Institute of Standards and Technology*) para unificar los términos, extraer las características más sobresalientes y comúnmente aceptadas, y elaborar un estándar, que fue aceptado por ANSI (*American National Standard Institute*) y publicado en febrero de 2004 bajo el código ANSI INCITS 359-2004, con el título de "Role-Based Access Control" [9].

El estándar no describe un modelo RBAC, sino que estructura su funcionalidad y características en forma de familia de modelos relacionados que, basándose en un núcleo que recoge la esencia de RBAC, proponen diferentes extensiones que pueden combinarse para dar lugar a modelos más ricos y ajustados a diferentes necesidades.

El estándar tiene dos grandes bloques: el **modelo de referencia RBAC** y las **especificaciones funcionales administrativas y del sistema RBAC**. El **modelo de referencia** define los conjuntos básicos de elementos RBAC y sus relaciones, estableciendo así el ámbito de las características RBAC soportadas. Las **especificaciones funcionales administrativas y del sistema** especifican las características requeridas para un sistema RBAC, divididas en tres categorías: operaciones administrativas, revisiones administrativas y funcionalidad del nivel de sistema.

El modelo de referencia RBAC se define en términos de tres componentes descritos brevemente a continuación:

- **`El núcleo de RBAC`**: recoge los elementos mínimos para poder hablar de RBAC, es decir, roles, usuarios, permisos y sesiones. Las relaciones esenciales son rol-permiso, que asocia permisos a roles, y usuario-rol, que asocia usuarios a roles. Los permisos se definen como asociaciones entre objetos y operaciones, sin entrar en la naturaleza de éstas ni de aquéllos, por lo que son vistos como símbolos sin significado para el modelo. El concepto de sesión relaciona usuarios con roles, permitiendo a los usuarios activar un subconjunto de los roles que tienen asignados, para su uso en una sesión concreta. Todas estas relaciones son de cardinalidad múltiple en ambos extremos, es decir, con semántica "de muchos a muchos".
- **`RBAC jerarquizado`**: añade relaciones para soportar jerarquías de roles, que permiten recoger las líneas de autoridad en la estructura de la organización. Las jerarquías definen una relación de herencia entre roles, de forma que el rol r_1 hereda de r_2 si todos los permisos de r_2 están presentes en r_1. En ese caso, se dice que r_1 es un rol *senior* y r_2 es *junior*. Se distinguen jerarquías generales, en las que se permite herencia múltiple, dando lugar a un grafo de roles, y jerarquías limitadas, en las que sólo puede haber herencia simple, dando lugar a estructuras en forma de árbol, más fáciles de manejar y ofreciendo en cualquier caso grandes ventajas administrativas con respecto al uso de roles sin jerarquizar.

- **RBAC con restricciones**: ofrece un mecanismo que da soporte al principio de separación de privilegios, permitiendo establecer relaciones de exclusividad entre roles. Cuando dos roles son mutuamente excluyentes, no pueden asignarse simultáneamente al mismo usuario. Se habla de **separación de derechos estática** si esta restricción se aplica en tiempo de asignación de usuarios, mientras que la **separación de derechos dinámica** tiene lugar en tiempo de activación de roles. Es decir, un usuario puede tener asignados dos roles mutuamente excluyentes, pero sólo podrá activar uno de ellos en una sesión. La separación de derechos puede ocurrir tanto en presencia de jerarquías como en ausencia de ellas. En el primer caso, las restricciones se heredarían a través de la jerarquía, multiplicándose el número de roles excluyentes.

En general, la ausencia de un permiso indica que el acceso no está permitido. Por ello, el estándar no contempla la existencia de permisos negativos, aunque se permite en las implementaciones. Puede lograrse el mismo efecto mediante el uso de restricciones.

Si bien las restricciones ofrecen una gran riqueza a la hora de expresar políticas de seguridad, siendo de hecho una de las principales motivaciones de usar RBAC, el estándar contempla solamente la separación de privilegios. Las restricciones son un área de investigación abierta, por lo que existen otros muchos tipos. Por ejemplo, la cardinalidad [10], que permite limitar el número de usuarios asignados a un rol; otras variantes de separación de derechos [11] como la separación de derechos histórica, que considera accesos pasados para restringir la aplicación de permisos en el futuro; la temporalidad [12], que permite imponer restricciones temporales (v.g. intervalos permitidos, duración máxima) a la existencia de un rol, de un permiso, de una asignación usuario-rol o permiso-rol o de una sesión; o las restricciones dependientes del contexto [13], que toman la forma de predicados construidos sobre valores de atributos de contexto, permitiendo recoger así el entorno en el que se produce la decisión de control de acceso.

7.4 Trabajo relacionado

La necesidad de modelos de seguridad específicos para hipermedia es ampliamente reconocida, debido en gran medida a las especiales características de este tipo de aplicaciones, que hace difícil e inapropiado adaptar modelos concebidos para sistemas de gestión de bases de datos o sistemas orientados a objetos, ya que no soportan componentes y servicios hipermedia. Si bien no existe un gran volumen de modelos abstractos de seguridad para hipermedia, sí hay más trabajos relacionados con la seguridad y el control de acceso en arquitecturas web, así como en tecnologías que se han ido desarrollando después como XML o los servicios web.

En los primeros modelos formales para hipermedia [13, 14] el modelado de la seguridad se basaba en controles de tipo DAC, con el objeto de garantizar la

confidencialidad. En otras palabras, pretendían responder sí o no a la decisión de control de acceso, sin entrar en qué podía realizar el usuario una vez se le concede acceso a un objeto.

En muchas organizaciones, los autores no son los verdaderos dueños de los objetos que producen, por lo cual es más adecuado el empleo de controles obligatorios (MAC). También existen modelos de seguridad de alto nivel para hipermedia, que permiten la definición de reglas de seguridad, pero no se relacionan con otras características propias de la hipermedia. Por ejemplo, el modelo de seguridad multinivel propuesto en [16] usa un mecanismo MAC que clasifica los objetos con los niveles de privacidad propios de la política militar.

Uno de los primeros trabajos exclusivamente relacionados con modelos de autorización e hipermedia es [17]. En él se establece claramente la necesidad de modelos específicos para este tipo de aplicaciones, y se discuten diferentes aspectos a considerar en el diseño de un modelo de autorización para hipermedia. Tras describir qué elementos conforman un modelo hipermedia, entre los que se incluyen los enlaces y los botones como principales mecanismos de interacción, se analiza la naturaleza de las autorizaciones en cada tipo de objeto.

Un trabajo posterior [18] define un modelo de autorización basado en las identidades de usuario, en el que se contemplan políticas de grano fino, en virtud de la diferenciación entre nodos y contenidos. Esto permite que diferentes usuarios tengan distintas vistas de un mismo nodo, la cual se genera en función de los permisos de usuario y de la ruta de navegación seguida hasta llegar al nodo. Respecto a las operaciones, se consideran manipulaciones propias de la hipermedia para modelar los permisos, denominados autorizaciones. Se tienen autorizaciones para navegar, para la autoría y para el uso, dividiéndose a su vez en tipos. Así, de las autorizaciones de navegación se tienen tipos de acceso para navegar (permite atravesar un enlace) y visualizar (diferentes partes o *slots* de un nodo dado). Entre las de autoría están la anotación, la creación de enlaces, la anexión de contenidos y la actualización (todas ellas referidas a los contenidos o *slots*). La autorización de uso permite incluir nuevos objetos en *slots* de un nodo. Por último, el modelo permite cierta propagación de autorizaciones entre objetos, organizando éstos en directorios, denominados dominios de autorización, y asignando privilegios sobre ellos. Este modelo no considera la posibilidad de incluir personalizaciones en el sistema.

En [19], que continúa con esta línea, se pone el énfasis en la naturaleza distribuida de la web, lo cual lleva a los autores a considerar extensiones al concepto de sujeto, incluyendo en el modelo a grupos de usuarios, RBAC o control basado en credenciales. Una **`credencial`** representa una serie de atributos de los sujetos cuyos valores deben ser acreditados por parte de los usuarios para poder disponer de ella [25]. Además, la representación de las autorizaciones y su cumplimiento pueden variar con objeto de ajustarse lo mejor posible a las necesidades derivadas del uso de arquitecturas descentralizadas (listas de control de acceso versus listas de capacidades).

Otro enfoque es el representado por mecanismos de RBAC, en los que el acceso se define en términos de los roles definidos en la organización. Este mecanismo es más eficiente en términos administrativos que los modelos DAC y MAC. Como se indica en [20] el paradigma de RBAC es un más que atractivo candidato para regular el acceso en aplicaciones web, debido a su potencial para soportar entornos multidominio. Este trabajo, además, resume el estado del arte en lo referente a modelos de control de acceso para aplicaciones web, incluyendo aspectos como tareas, flujos de trabajo (*workflows*) o certificados.

El primer intento de integrar mecanismos RBAC en la web se remonta a [21], como continuación de trabajos previos iniciados en 1997. Este trabajo describe en detalle el modelo RBAC del NIST, recientemente aceptado como estándar [9] y comentado en la sección 7.3.4, así como la implementación para servidores web en intranets, denominada RBAC/Web. Esta propuesta se centra puramente en la tecnología web del momento, por lo cual no hay relación con conceptos más abstractos del dominio de la hipermedia. Además, los objetos protegidos son vistos como un todo sin partes ni relaciones estructurales ni de navegación, por lo que es imposible considerar políticas de grano fino que permitan diferentes vistas de la misma información.

En [22] se propone un enfoque RBAC descentralizado para sistemas hipermedia cooperativos, donde los roles se complementan con la noción de equipos, definidos como un conjunto de usuarios con un objetivo común. El autor de un objeto se le considera su dueño, que discrecionalmente define las reglas de acceso para el mismo.

En los últimos años, con el auge de XML [23] como metalenguaje para la descripción de documentos, han surgido modelos de control de acceso dirigidos a documentos en este formato. Aunque se centran en esta tecnología y por tanto no consideran abstracciones hipermedia, su relevancia es clara ya que XML es una plataforma muy adecuada para representar elementos hipermedia.

Los trabajos relacionados con este área se centran en proteger de forma genérica un documento XML. Se parte de la premisa de que diferentes partes de un mismo documento tienen diferentes requisitos de protección, por lo que es necesario controlar el acceso selectivamente, aprovechando la estructura jerárquica de XML, logrando así acceso de grano fino. Algunos de ellos también ponen énfasis en definir arquitecturas que den soporte a los modelos propuestos, aunque no se mencionan en este capítulo, ya que no están relacionadas con los objetivos del mismo.

Los primeros trabajos se encuadraban en dos líneas de investigación, encabezadas por Damiani [24] y Bertino [25]. Estos trabajos protegen definiciones de tipos de documento (DTD, *Document Type Definition*) así como instancias XML, asumiendo un modelo DAC. En [20], los permisos se expresan mediante tuplas <sujeto, objeto, permiso, recursividad>, donde sujeto representa identidades de usuarios o la ubicación de éstos (v.g. dirección IP); objeto es una expresión XPath [26], que permite obtener un subconjunto de elementos de un documento XML; permiso puede ser positivo o negativo, indicando si el sujeto puede leer o no el objeto;

y recursividad expresa cómo el permiso se ha de propagar hacia, bien las instancias, bien hacia los elementos contenidos dentro del objeto.

En [25] se introduce la noción de credencial para establecer de forma más flexible las autorizaciones. Al suponer control de acceso en entornos web, la identidad de los usuarios no es conocida a priori, por lo que se utilizan credenciales que reflejan in conjunto de propiedades arbitrarias de un sujeto, ya sean personales o derivadas de relaciones con otros sujetos (v.g. cualificaciones en una organización). Los objetos protegidos pueden ser DTDs o instancias XML, representables de forma similar a como lo hace XPath, y aplicándose los principios de propagación ya mencionados en otros trabajos. Se soportan dos tipos de permisos: navegación y autoría. Los permisos de navegación se subdividen con objeto de permitir sólo la visualización de un documento o bien de éste y todas sus referencias a otros, mientras que los de autoría hacen otro tanto para distinguir entre adiciones a un elemento sin modificar su contenido y modificaciones de cualquier tipo sobre un elemento.

En [27] se propone un modelo RBAC extendido para realizar control de acceso sobre objetos XML. Puede regularse el acceso bien a nivel de esquema (*XML Schema*) [28], de forma que los permisos expresados para esquemas se aplicarán sistemáticamente a todas las instancias que se creen conforme al mismo, o bien puede regularse a nivel de instancia (documento XML), lo que permite tratar de forma diferente a diferentes instancias de un mismo esquema. Respecto a los sujetos, se organizan en roles de acuerdo con el modelo que posteriormente serviría de base para el estándar RBAC [9]. Una aportación relevante de este trabajo es la noción de reutilización de permisos, en virtud de una jerarquía establecida a nivel de esquemas definida mediante relaciones de derivación de tipos definidos por el administrador. Esto permite reducir drásticamente la administración de autorizaciones, al permitirse la herencia a través de la jerarquía de roles, y la reutilización a través de la jerarquía de objetos. Además, un permiso a nivel de esquema se puede propagar recursivamente hacia otros esquemas contenidos en él, permitiéndose la especificación del número de niveles recursivos que se desean propagar. El modelo permite operaciones de creación, lectura, modificado y borrado.

En [29] se asume un modelo basado en identidad, en el que los objetos protegidos son DTDs e instancias XML expresadas mediante XPath. También se contempla la idea de propagar permisos definidos en la DTD hacia sus instancias, pudiendo expresarse operaciones de lectura, modificación y borrado.

El lenguaje XML no sólo sirve como objeto del control de acceso, sino también permite expresar políticas de seguridad de una forma independiente de la aplicación y por tanto, interoperable. En este sentido, los trabajos más relevantes son las especificaciones SAML (*Security Assertion Markup Language*) y XACML (*eXtensible Access Control Markup Language*). SAML [30] define un marco de trabajo basado en XML para representar e intercambiar información de autentificación y autorización entre entidades de negocio, y confía en terceras partes (autoridades de confianza) que proveen las aserciones que contienen esa información. Una aserción indica unas determinadas características de una entidad, bien referentes

a su identificación (v.g. período de validez), a ciertas propiedades (v.g. el usuario es cliente preferente), o bien a autorizaciones a las que tiene derecho (v.g. comprar determinados artículos). XACML [3] es otro marco de trabajo basado en XML, pensado para especificar políticas de control de acceso para recursos web. Permite expresar requisitos de control de acceso basado en identidad, e incluye la posibilidad de indicar parámetros de contexto, esto es, representar elementos como el tiempo o la ubicación, incorporándolos en la decisión de control de acceso. En todo caso, XACML no se diseñó sobre un modelo formal de control de acceso sensible al contexto, ni basado en roles. Existe un perfil XACML para RBAC, en el que se define cómo expresar una política RBAC usando XACML, pero no contempla el concepto de sesión, por lo cual todos los roles están activos en todo momento para el usuario.

Por ultimo, los servicios web [31] representan el siguiente paso en el paradigma de la web: en un principio el objetivo estaba en la conectividad (TCP/IP), después en la presentación (HTML) para pasar a la capacidad de ser programable (servicios web). Los servicios web permiten la interoperabilidad en entornos distribuidos y heterogéneos, y por tanto, son un buen candidato para soportar aplicaciones hipermedia abiertas. Aunque fuertemente influenciados por la tecnología XML, ya que los estándares de servicios web se expresan en XML como forma de promover la interoperabilidad, existen aspectos de control de acceso que deben ser tratados en este tipo de entornos.

En concreto, derivado de la arquitectura distribuida y heterogénea, se hace preciso ampliar el contexto en el que se produce el control de acceso [32] mencionado en la introducción de este capítulo, para incluir elementos como:

- El acceso a un servicio puede ser solicitado desde múltiples lugares, cada uno de los cuales tendrá definido un determinado contexto (por ejemplo, una ubicación, la identidad de la empresa que solicita el servicio, etc.)
- Dejando de lado asuntos de confianza en la información (*trust management*), no necesariamente se conoce de antemano el usuario que desea acceder, de manera que no es posible saber qué roles, de los definidos para el servicio, puede adoptar. Por ello, una alternativa interesante sería considerar el uso de credenciales de usuario, como mecanismo intermedio para automatizar la asignación de usuarios a roles en función de la capacidad de aquéllos para acreditar ciertos atributos requeridos para pertenecer al rol.
- Puede haber restricciones temporales que inhabiliten el servicio en sí, o partes de éste, a determinados sujetos en determinados intervalos de tiempo.

En general, estos nuevos desafíos requieren incorporar restricciones dependientes del contexto al modelo de control de acceso.

Pero la seguridad en servicios web no se limita al control de acceso. La posibilidad que ofrecen los servicios web para enlazar aplicaciones, tanto dentro de las organizaciones como fuera de ellas, de una forma interoperable y neutral, ha situado los asuntos de seguridad en primer plano. El conjunto más notable de especificaciones que se está desarrollando está descrito en el *Web Services Security Roadmap* [33]. Cabe mencionar la especificación *WS-Security* [34], que ofrece confidencialidad e

integridad a nivel de mensaje en la comunicación que se produce durante la prestación del servicio. *WS-Policy* [35] permite describir políticas de seguridad en términos de sus características (v.g. requerimiento de un determinado *token* de seguridad, uso de un algoritmo de cifrado), que después pueden asociarse a los diferentes elementos que componen un servicio web. *WS-Trust* [36] define un modelo de confianza que permite el intercambio de *tokens* de seguridad, para habilitar la emisión y diseminación de credenciales entre diferentes dominios de seguridad, estableciendo relaciones de confianza.

En el ámbito del diseño de la seguridad del sistema, también puede considerarse el uso de **patrones de seguridad**. Un patrón representa una solución bien conocida ante un problema recurrente que aparece en un determinado entorno. En la parte III de este libro se ofrece un amplio tratamiento de los patrones, por lo que ahora sólo se describe el uso de patrones en el diseño de la seguridad de sistemas web. En el contexto de la seguridad, los patrones encapsulan la experiencia en forma de soluciones dirigidas a problemas de seguridad, destacando los principales aspectos derivados de su aplicación [37]. Un patrón de seguridad es ligeramente diferente de lo que se conoce como patrón de diseño, si bien la mayoría de ellos pueden verse como tales ya que proponen soluciones que tienen un reflejo en la implementación del sistema. No hay una definición única y consensuada del concepto de patrón de seguridad, pero existe un proceso de consenso general en la comunidad que investiga en este área.

Se distinguen dos categorías generales de patrones de seguridad [37]:

- Patrones estructurales: se asemejan a los patrones de diseño, ya que pueden ser implementados en el sistema final. Típicamente incluyen diagramas de secuencia y de interacción
- Patrones procedimentales: pueden utilizarse durante el desarrollo de software crítico para la seguridad. Típicamente tienen impacto en la organización o en la gestión del proyecto

Uno de los problemas en los que se está trabajando es en la elaboración de una plantilla para patrones, que define un formato consistente para describir un patrón. Uno de los campos que diferencia este tipo de patrones de otros es el de "aspectos a considerar" (*issues*), en el que se explican los errores más comunes que ocurren en el uso del patrón, así como las sugerencias para evitarlos. También se enumeran las vulnerabilidades derivadas de su uso, indicando las circunstancias bajo las cuales una buena implementación del patrón puede ser atacada. Por último, se comentan posibles ataques contra el patrón como tal, estableciendo las defensas contra los mismos. Otro campo es el de "consecuencias" (*trade-off*), en el que se describe el impacto del uso del patrón respecto a los requisitos, tanto funcionales como no funcionales. No se usa para describir efectos secundarios, sino sólo efectos directos del uso, en una de las siguientes áreas: contabilidad, disponibilidad, confidencialidad, mantenibilidad, usabilidad, rendimiento y coste. Para ilustrar el concepto de patrón de seguridad, considérense a modo de ejemplo el patrón **Aserción Segura** [37] y los relacionados con modelos de seguridad [38].

El patrón **`Aserción Segura`** reparte comprobaciones de seguridad dependientes de la aplicación a lo largo de todo el sistema. Estas comprobaciones se denominan aserciones, y sirven para asegurar el cumplimiento de asunciones realizadas por los programadores respecto al entorno y al comportamiento del sistema. Un sistema de detección de intrusos puede partir de esas aserciones para detectar y correlar problemas a nivel de aplicación que usualmente indican intentos de uso fraudulento del sistema.

Los patrones relacionados con modelos de seguridad propuestos en [38,39] permiten poner en práctica varios modelos de seguridad, entre los que se encuentra RBAC. Todos ellos se basan en modelar mediante objetos los diferentes elementos del modelo de seguridad (v.g. sujetos, objetos, asignaciones...), y proponen diseños que solucionan problemas típicos, tales como la implantación de RBAC como tal [38,39], la aplicación del principio de menor privilegio, o el uso de jerarquías de roles [39].

Los patrones de seguridad tienden un puente entre el diseño de la seguridad y el diseño del resto del sistema, por lo que son de gran ayuda para considerar la seguridad en todas las etapas de diseño, y articular más fácilmente la política de seguridad.

7.5 Modelado del acceso en hipermedia (MARAH)

Esta sección presenta un modelo de control de acceso para aplicaciones hipermedia denominado MARAH (Modelo de Acceso basado en Roles para Aplicaciones Hipermedia). La principal característica de este modelo es que trabaja con conceptos del dominio de la hipermedia, lo que permite a los diseñadores modelar el acceso al sistema al mismo tiempo que modelan el sistema en sí. MARAH está basado en la filosofía RBAC por las ventajas que este mecanismo proporciona, especialmente valuables en entornos hipermedia debido entre otras cosas al gran número de usuarios que potencialmente pueden acceder a él, y a la naturalidad con la que refleja la estructura organizativa.

En primer lugar se presentarán los niveles de abstracción que se han considerado en el diseño del modelo, para después introducir los diferentes componentes de MARAH, acompañados de ejemplos que ilustran su utilidad.

7.5.1 Niveles de abstracción

A la hora de acometer el diseño de un modelo de acceso para hipermedia, surge la necesidad de encuadrarlo en un contexto de abstracción que permita definir el tipo de conceptos que se van a manejar en el modelo. Esta decisión es de crucial importancia de cara a proporcionar un modelo adecuado, tanto para los diseñadores del sistema como para la aceptación por parte de los usuarios.

Considerando un sistema hipermedia desde este punto de vista, surgen tres niveles de abstracción diferentes, tal y como muestra la Figura 7.1:

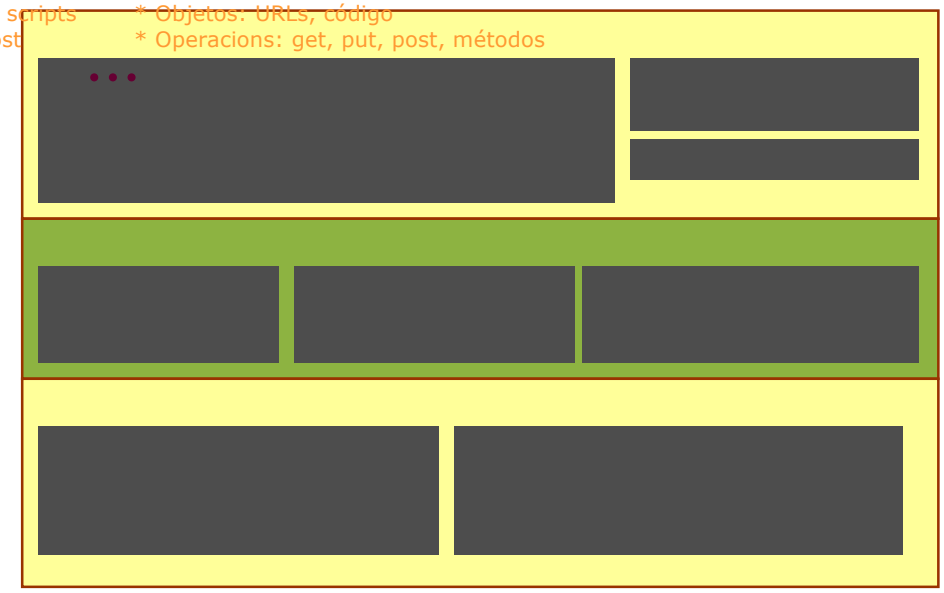


Fig. 7.1: Niveles de abstracción en el diseño de MARAH

Nivel de Aplicación

Este es el nivel percibido por los usuarios del sistema. En él se considera la aplicación como una herramienta en el contexto de una organización, de forma que ofrece a sus usuarios un conjunto de operaciones de alto nivel relacionadas con el dominio de aplicación en el que la herramienta es útil, que les permite realizar parte de su trabajo cotidiano. Por ello, a este nivel se considera la estructura organizativa, basada en la definición de roles de usuario y equipos de trabajo, y las operaciones realizables con el sistema, en términos de funciones de trabajo o actividades de un **`flujo de trabajo`** (*workflow*). El control de acceso sería un mecanismo que determinaría si un determinado usuario puede o no realizar una función del sistema, es decir, acometer una tarea del workflow, sin preocuparse de los objetos concretos accedidos por la operación.

El estándar de RBAC [9] mencionado en la sección de diseño de políticas de seguridad trabaja a este nivel de abstracción, ya que pone el énfasis en el concepto de rol como intermediario entre usuarios y permisos, y define los permisos como símbolos (*tokens*) no interpretados, abstrayéndose así de su naturaleza concreta, cuya existencia aprueba una determinada interacción del usuario con el sistema. En esta línea, los trabajos relacionados con el control de acceso para workflows [40] tratan de restringir a ciertos sujetos las acciones correspondientes a la activación, ejecución, detenimiento o finalización de tareas.

Este nivel es muy adecuado desde el punto de vista del administrador de seguridad, ya que la definición de las reglas de acceso se realiza en los mismos términos, o muy similares, que la política de acceso, en tanto que se utiliza directamente la estructura organizativa y la protección se centra en acciones cuya semántica es muy comprensible. Por tanto, en su forma más simple, los permisos se expresarían como tuplas <rol, operación>.

El problema que surge en este nivel es que la implementación del modelo es completamente dependiente de la aplicación donde se aplica, ya que una vez definidas las reglas sería preciso descender al código para implementar los controles necesarios, haciendo imposible la reutilización del monitor de referencia en otra aplicación. Otras alternativas, basadas en el uso de modelos de seguridad para workflows [40], dejarían de lado completamente aspectos importantes para la hipermedia, como las políticas de grano fino, ya que se centrarían en permitir/impedir la ejecución de una tarea, por tanto sigue siendo necesario descender para determinar qué elementos funcionales y de datos son afectados por una decisión de control de acceso.

Nivel Físico

Este nivel considera aspectos puramente tecnológicos relacionados con la plataforma que da soporte al sistema, muy cercanos a la implementación del mismo, como por ejemplo ficheros, protocolos de red o servidores. Así, el acceso al sistema se definiría en términos de operaciones sobre archivos, como crear, leer o escribir, o, más relacionado con la web, con operaciones sobre servidores web mediante el protocolo HTTP aplicadas sobre URLs, tales como `GET index.html`. También se podrían incluir aquí las abstracciones de seguridad proporcionadas por las tecnologías empleadas para programar sistemas web en el lado del servidor, como por ejemplo Java, PHP, Zope o ASP.NET. Si bien existen diferencias respecto al concepto de sujeto en función del modelo de seguridad ofrecido por la tecnología, para los objetos en el mejor de los casos trabajarían a nivel de protección del código, previniendo a ciertos usuarios la ejecución de determinados métodos o fragmentos de código.

En un modelo de seguridad definido en este nivel, los permisos se expresarían como tuplas <sujeto, objeto, operación>, de forma que el modelo resultante es completamente independiente de la aplicación a proteger, y, por tanto, es flexible y reutilizable en diferentes dominios.

Pero por otro lado, precisamente esa diferente conceptualización de los sujetos para diferentes tecnologías (básicamente, usuarios/grupos o roles con ciertas limitaciones) produce una necesidad de adaptar las abstracciones del nivel de aplicación a la concepción concreta usada en la plataforma de implementación. Por otra parte, es difícil establecer una correspondencia clara y unívoca entre funciones de alto nivel identificadas en el diseño y objetos del servidor que las implementan de acuerdo a una estructura hipermedia concreta, ya que hay un hueco conceptual demasiado grande entre ambas formas de diseño. Adicionalmente, suele tenerse una visión monolítica del objeto a acceder, cuando lo deseable sería considerar la unidad de acceso a la

información como un elemento compuesto de partes, algunas de las cuales podrían ser accesibles y otras no.

Soportar esta característica requeriría, bien el uso de tecnologías como XML, que soporta inherentemente la composición jerárquica, bien implementarla ad hoc para un sistema concreto, lo cual volvería a dificultar la reutilización. En cualquier caso, no es trivial el soporte para control de acceso respecto a otras características de la hipermedia no contempladas en los modelos de programación orientados a objetos en que se basan las tecnologías actuales. Por ejemplo, la ubicación de contenidos en un nodo (v.g. espacio/temporal, como se menciona en el capítulo 2) o la abstracción que permite generalizar tipos de nodos de acuerdo a unas características comunes.

Nivel Hipermedia

Este es el nivel intermedio, en el que se ubica MARAH. En este nivel, la aplicación se concibe como una instancia de un modelo hipermedia, en la que ya se han tomado decisiones respecto a la estructura y navegación de la información, pero aún no se ha implementado en un sistema concreto. De esta forma, en lo referente a objetos ya se dispone de un diseño concreto o modelo del sistema, y respecto a los sujetos se tiene la misma estructura basada en roles que había en el nivel de aplicación. Este nivel es un buen candidato para ubicar el modelo control de acceso, ya que:

- Permite expresar los permisos en la forma <rol, objeto, operación>, de manera que se tiene un mecanismo flexible e independiente de la aplicación
- No se depende de una plataforma concreta de implementación
- El modelo es abstracto y reutilizable ya que usa conceptos del dominio de la hipermedia, como son el nodo, contenido o enlace, para expresar las reglas de acceso (principio de abstracción)

Las operaciones en este nivel son menos sencillas que sus correspondientes en el nivel físico, y a su vez, más reutilizables que las del nivel de aplicación, ya que no dependen de un dominio concreto, en otras palabras, las operaciones del nivel de aplicación pueden expresarse en forma de operaciones de un modelo hipermedia, las cuales a su vez tendrán un reflejo en alguna plataforma de programación.

Los objetos en este nivel recogen toda la riqueza de la aplicación hipermedia, son más genéricos que sus homólogos del nivel físico, y permiten la especificación de estructuras de navegación y relaciones de agregación y generalización. Además, estas relaciones no sólo sirven para modelar la aplicación, sino también para gobernar de alguna forma la propagación de privilegios, como se verá más adelante.

En resumen, de igual forma que un modelo hipermedia recoge las características de diferentes aplicaciones y debe ser implementado en una determinada tecnología, el modelo de control de acceso sigue esta filosofía, permitiendo así otorgar a la seguridad el papel que se merece dentro del desarrollo al disponerse de mecanismos de modelado integrados con el resto de aspectos del sistema.

7.5.2 Modelado de sujetos

En MARAH los sujetos son entidades capaces de iniciar una operación sobre un objeto. Tradicionalmente se distingue entre usuario (persona) y sujeto (programa que actúa en nombre de una persona), aunque a efectos del modelo esta distinción es irrelevante. El modelo asume que, dada una petición de acceso, existe y se conoce de forma fiable un usuario autenticado en cuyo nombre ésta se produce.

Los sujetos se estructuran en *roles* que representan funciones organizativas. Por su propia definición, los roles pueden ser más o menos concretos, lo que da lugar a relaciones de **`generalización`** que permiten definir jerarquías de roles de acuerdo a la semántica *es-un*. Por ejemplo, un profesor ayudante es un profesor, y un profesor titular también. En este sentido, los roles más específicos tendrían más privilegios (representarían funciones muy concretas), correspondiéndose casi por completo con los roles *senior* de RBAC, mientras que los más generales englobarían los permisos por defecto, y por tanto serían los roles *junior*. Los roles senior heredan los permisos de los *junior*, de la misma forma indicada en el estándar RBAC [9].

En ocasiones es interesante agrupar un conjunto de roles, bien para formar un grupo de interés (v.g. diseñadores de PHP), bien para recoger la estructura de la organización (v.g. departamento de matemáticas), o simplemente para simplificar tareas administrativas (v.g. usuarios del sistema). Por ello MARAH ofrece el concepto de **`equipo`**, que permite considerar un conjunto de roles heterogéneos como una entidad organizativa, en virtud de relaciones de **`agregación`** con semántica *todo-parte*. Por ejemplo, un grupo de investigación puede componerse de personas con el rol de profesor ayudante, catedrático, director de grupo e investigador asociado. Mientras que la generalización agrupa roles homogéneos, la agregación hace lo propio con roles heterogéneos. En ambas relaciones, suele denominarse elemento *padre* al que agrega o generaliza (análogo al junior), e *hijo* al agregado/generalizado (análogo al senior).

Los usuarios se asignan a roles mediante la relación de asignación, idéntica a la ofrecida en modelos RBAC. Nótese que, si bien tiene sentido asignar usuarios a roles generales (v.g. porque tengan menos privilegios), no tiene sentido asignar usuarios a equipos, ya que la semántica del equipo sólo define la agrupación como tal, y no un papel específico dentro de la misma. De hecho es difícil concebir que un usuario esté cualificado para realizar a la vez todas las funciones que un equipo engloba. Por ello, un usuario no puede formar parte de un equipo directamente, sino a través de algún rol que forme parte del mismo.

Es importante resaltar que el modelado de sujetos en MARAH es diferente del usado en RBAC. En concreto, si bien RBAC no contempla el concepto de equipo explícitamente, es sencillo encontrar en la literatura ejemplos de roles usados a tal efecto, normalmente con el propósito de especificar permisos generales que todos los usuarios pertenecientes al equipo deben tener. Por este motivo se ha decidido diferenciar explícitamente el concepto de equipo. En todo caso, para referirse indistintamente a roles y a equipos, MARAH emplea el término de sujeto. Por otra

parte, en términos matemáticos, las jerarquías soportadas por MARAH se corresponden con dos órdenes parciales de idénticas propiedades a los definidos en el estándar RBAC [9] para modelos con jerarquías generales, es decir, se tienen dos estructuras en forma de grafo acíclico dirigido (GAC), en el que puede haber herencia múltiple (un rol puede ser generalizado y/o agregado por varios).

Para ilustrar el modelado de sujetos, considérese la Figura 7.2. En ella se recoge la estructura organizativa de un hipotético equipo de desarrollo. Los roles se representan con cajas en las que aparece una cabeza, mientras que los equipos son cajas con dos cabezas. Las relaciones de agregación se representan con rombos dibujados en el elemento padre, mientras que las generalizaciones son triángulos.

Puede verse cómo el Equipo de Desarrollo agrega 4 sujetos, tres de los cuales son roles y uno representa el equipo de usuarios, que a su vez se compone de dos roles, el usuario y el coordinador. Por otra parte, el rol programador se especializa en programador de interfaz y programador de bases de datos, de forma que éstos son roles *senior* respecto al programador. Es importante observar que no tiene sentido que en las hojas del árbol resultante haya equipos. Cuando se trata de una estructura en grafo, los nodos que no actúen como padre han de ser necesariamente roles. A la hora de asignar usuarios, éstos pueden asignarse a cualquier rol, independientemente de si es *junior* o *senior*, pero nunca a equipos.

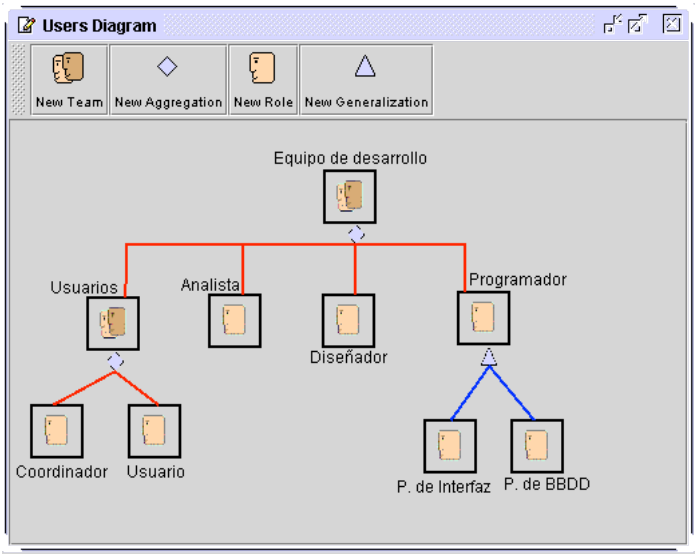


Fig. 7.2: Ejemplo de modelado de sujetos en MARAH

7.5.3 Modelado de objetos

Al ser MARAH un modelo de seguridad para hipermedia que utiliza los conceptos propios de esta disciplina, los *objetos protegidos* son los elementos de un diseño hipermedia. Si bien en principio no hay razón para pensar que MARAH sólo es aplicable a un modelo hipermedia concreto (ya que muchos de sus componentes son independientes), el desarrollo del modelo de acceso se realiza asumiendo un modelo hipermedia subyacente que proporciona las abstracciones adecuadas. En el caso de MARAH se trabaja sobre el modelo de referencia Labyrinth [41], que se compone de los siguientes elementos: nodos, contenidos, anclas, enlaces, atributos y eventos.

Los **`nodos`** son contenedores abstractos de información, que representan una entidad del dominio de aplicación cuya existencia tiene sentido por sí misma. Los **`contenidos`** son piezas de información, que se asocian a los nodos mediante una relación de ubicación. Las **`anclas`** son áreas situadas en nodos o contenidos, definidas en términos de las unidades de representación del elemento al que van asociadas (v.g. caracteres, píxeles, segundos, fotogramas, escenas, etc.). Los **`enlaces`** relacionan un conjunto de anclas de origen con otro de destino, estableciendo conexiones entre nodos o contenidos. Los **`atributos`** permiten anotar cualquier elemento del modelo con información, normalmente relacionada con propiedades del elemento que interesa representar, para así incrementar su semántica. Los **`eventos`** son piezas de código que se ejecutan cuando una determinada condición se verifica, y van asociadas a nodos, contenidos o enlaces, de forma que la condición se comprueba cuando el elemento asociado está activo. Permiten representar especificaciones procedimentales del sistema, así como comportamientos interactivos.

Así, puede decirse que los objetos protegidos de MARAH son los elementos de Labyrinth. De todos estos elementos, hay que descartar las funciones del modelo, que permiten relacionar elementos de un conjunto con elementos de otro. Así por ejemplo, la función de ubicación permite relacionar contenidos con respecto al nodo donde se inscriben. Además de las funciones, MARAH deja de lado a los usuarios y a los grupos, que forman parte del modelo también, por dos motivos: características ofrecidas por estos elementos van a ser sustituidas por las que ofrece el modelo de acceso (de hecho, MARAH se concibió en un principio a raíz de estos elementos), y en segundo lugar, proteger estos elementos supone incluir en el control de acceso a las tareas administrativas. Si bien RBAC puede usarse para administrar RBAC, controlando la forma de gestionar los roles, usuarios y permisos, la versión actual de MARAH no da soporte a la administración.

De esta forma, la discusión se reduce a proteger los conjuntos de Nodos, Contenidos, Anclas, Enlaces, Atributos y Eventos. De estos elementos, los atributos van ligados indisolublemente a cualquier otro elemento del modelo, mientras que ocurre otro tanto con los eventos y los nodos, contenidos o enlaces. Por ello, una suposición inicial, basada en el principio de sencillez, es que atributos y eventos toman las autorizaciones que se definan para el elemento al que van asociadas. Esta suposición

considera que no tiene sentido interactuar a uno de estos elementos por sí solo, sino a través del nodo, contenido o enlace asociado. De una forma similar, se asume que las anclas toman las autorizaciones del contenido o nodo en que se definen, de manera que un ancla será accesible para quienes puedan acceder al elemento en que se haya definido. Por consiguiente, se tendrá acceso a un enlace si al menos puede accederse a algún ancla de origen y a una de destino.

Dicho todo esto, se tiene que los objetos a proteger son nodos y contenidos. En un principio bastaría con disponer de sus identificadores para tener una base sobre la que establecer permisos. Pero Labyrinth permite definir para ambos elementos relaciones de agregación y generalización. La generalización permite definir elementos más genéricos, con la idea de agrupar propiedades comunes. Por ejemplo, puede hacerse que todos los nodos incluyan un determinado contenido (v.g. un logo) sin más que añadirlo a un nodo más genérico que generalice a todos los demás. Este mismo mecanismo haría que los contenidos pudieran heredar atributos que los caractericen o eventos que definan su comportamiento.

La agregación permite agrupar elementos heterogéneos para poder tratarlos como un todo. Por ejemplo un nodo que represente un libro puede agregar nodos que representen sus capítulos. La agregación es una relación puramente estructural, no debe confundirse con la inclusión de contenidos en nodos, ni con la existencia de enlaces entre nodos. Aparte de para recoger la estructura del hiperdocumento, estas relaciones sirven para propagar características de unos nodos o contenidos a otros, cuando éstos se refieren a conceptos distintos (v.g. la inclusión de un logo en todos los nodos).
Sobre estos mecanismos de composición, se define el concepto de **`dominio`**, que como se verá después tiene implicaciones en la definición de la política de acceso. El dominio de un objeto *o* está formado por tres elementos: el propio *o*, el dominio de todos los objetos que son agregados por *o* y el dominio de todos los objetos generalizados por *o*. Así, se tiene que el dominio de un objeto son todos los objetos que de alguna forma u otra son "alcanzables" a partir del objeto siguiendo relaciones de agregación y/o generalización. De esta forma, al referirse a un dominio, se engloba un conjunto de objetos (bien nodos, bien contenidos) mediante un único nombre, simplificando así la gestión de las autorizaciones, como se verá más adelante.

La Figura 7.3 muestra un ejemplo de modelado de objetos para una aplicación hipermedia relacionada con una biblioteca. Por simplicidad, sólo se muestran los nodos. El nodo que representa a la biblioteca, que podría corresponderse con la página principal al entrar en la aplicación, se compone de tres nodos, que representan las fichas de usuarios, los documentos existentes en la biblioteca, y el catálogo OPAC de consulta. Debe recalcarse que esta agregación no implica nada respecto a la presentación de la información, solamente refleja relaciones estructurales, si bien sirven para propagar propiedades comunes. A su vez, los documentos se especializan en revistas y libros. El motivo de esta división podría radicar, entre otras cosas, en los contenidos que se incluyen en cada nodo, que, además de los heredados por el hecho de ser documentos, pueden incluir elementos especializados (v.g. el volumen y el número, en el caso de las revistas). Los nodos también pueden representar espacios

donde se ofrece una determinada funcionalidad, como por ejemplo el buscador de documentos.

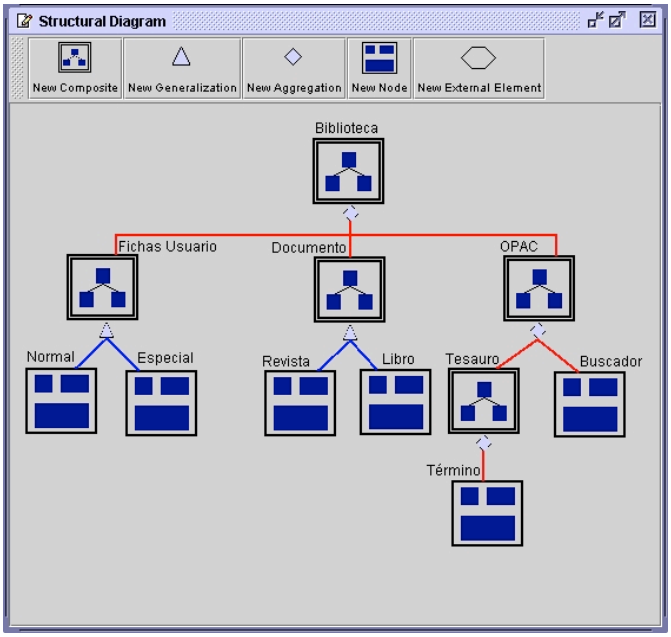


Fig. 7.3: Ejemplo de modelado de objetos en MARAH

7.5.4 Modelado de permisos

El estándar de RBAC define un permiso como una aprobación para realizar una operación sobre uno o más objetos protegidos [9]. En ningún momento se entra en la naturaleza de los permisos, ya que ésta es dependiente del sistema a proteger. El estándar ofrece funciones matemáticas para, dado un permiso, obtener los conjuntos de objetos y operaciones a los que se refiere.

En MARAH, una vez definidos los objetos, hay que definir las operaciones que pueden aplicárseles. Debido a que se fundamenta en Labyrinth, las operaciones de MARAH son tomadas directamente del conjunto de operaciones definidas en Labyrinth [42]. Si bien Labyrinth define operaciones administrativas, que permiten tanto la creación como el mantenimiento del hiperdocumento, MARAH se centra en las operaciones propias del *tiempo de uso* del sistema, aquél en el que se navegan los nodos, se crean versiones personalizadas de éstos o se modifican sus contenidos. Es por ello que las operaciones que conforman los permisos son del tipo `activarEnlace`, `ejecutarEvento`, `obtenerContenidosNodo` o leerNodo.

Para no tener que definir explícitamente qué operaciones se incluyen en un determinado permiso, se introduce el concepto de **categoría de seguridad**, que se define como una etiqueta que representa un tipo abstracto de manipulación que puede darse en cualquier sistema hipermedia. Estas etiquetas se estructuran en forma de orden parcial, en el que un valor añade capacidades de manipulación sobre el anterior. Las categorías definidas en MARAH son:

- **Navegación**: representa la capacidad de recuperar información del sistema, por ejemplo utilizando un motor de búsqueda, o accediendo directamente al elemento mediante su identificador.
- **Personalización**: añade a la navegación la capacidad para crear versiones personalizadas de un objeto por parte de un usuario o un conjunto de éstos. Así, un determinado elemento (o mejor, una copia privada de éste) podrá modificarse y ser accedido después por un parte de los usuarios.
- **Edición**: añade a la personalización la posibilidad de modificar elementos del hiperdocumento. Estos cambios serán percibidos por todos los usuarios.

Las categorías de seguridad representan modos de acceso al sistema, de manera abstracta. Como se va a comentar posteriormente, se utilizan para clasificar muchos elementos del modelo de seguridad. Nótese que, a pesar de definirse un orden parcial, las categorías contempladas en la versión actual de MARAH están totalmente ordenadas, de manera que no hay categorías incomparables.

La ejecución de las operaciones de uso definidas en Labyrinth involucra capacidades de manipulación propias de una de estas tres categorías. Por ello, se define la relación de **clasificación de operaciones** como una asociación entre una operación y la categoría de seguridad que representa el tipo de manipulación mínimo necesario para ejecutarla. De esta forma, operaciones como `crearNodo` o `definirAncla` se asocian a la categoría de Edición, mientras que `leerNodo` u `obtenerContenidosNodo` requieren habilidades de Navegación. Se aplica el principio de mínimo privilegio, ya que no se asigna a las operaciones un tipo de manipulación más permisivo de lo estrictamente necesario.

Con lo visto hasta ahora, MARAH dispone de un mecanismo para referirse a un conjunto de objetos (el dominio) y un mecanismo para referirse a un conjunto de operaciones (la clasificación de operaciones). Al igual que un permiso en RBAC se define como una relación entre un conjunto de operaciones y un conjunto de objetos, MARAH permite definir autorizaciones, denominadas **acreditaciones**, cuya semántica es idéntica a la de los permisos RBAC: ambos conceptos se definen de igual forma, dado cualquiera de ellos es posible determinar qué operaciones y qué objetos involucra, y por último las acreditaciones son *positivas*, es decir, expresan capacidades para hacer algo, y no prohibiciones. La diferencia con los permisos RBAC es que no puede configurarse cualquier combinación de objetos/operaciones, ya que las categorías de seguridad forman un orden. En otras palabras, no puede darse una acreditación que incluya una operación clasificada con la categoría de Edición y que no incluya una operación clasificada con la de Navegación, ya que la Edición incluye a la Navegación. Al igual que los permisos, las acreditaciones serán asignadas a los roles.

Hasta este momento, en cuanto a permisos se refiere, se tiene que MARAH es una interpretación del modelo genérico de RBAC, es decir, define la naturaleza de los permisos especificando las operaciones y los objetos como elementos de un modelo hipermedia. Pero además de las acreditaciones, existen más mecanismos para definir autorizaciones. Si bien ha quedado demostrado que RBAC permite simular el comportamiento de modelos DAC y MAC [4], MARAH dispone de elementos que se asemejan en cierta forma a estos modelos, y que pueden coexistir con RBAC si se considera necesario. En concreto, pueden definirse permisos negativos discrecionalmente como ocurre en DAC, en forma de **`listas de control de acceso negativas`** (nACL, *negative access control list*) que almacenan sujetos que tienen prohibido el acceso a un determinado dominio, y establecer clasificaciones de seguridad en los objetos (MAC). Estos mecanismos son complementarios, y su uso está pensado para aportar mayor flexibilidad en la expresión de una política de seguridad así como facilitar la gestión.

Asignación de autorizaciones

Como se acaba de comentar, MARAH define tres mecanismos para asignar autorizaciones al definir la política de seguridad.

El primero de ellos es la **`función de acreditación`**, que relaciona sujetos (roles o equipos) con permisos, teniendo así tuplas <sujeto, dominio(o), categoría>. Este mecanismo está orientado a garantizar la integridad del hiperdocumento, ya que determina qué usuarios pueden realizar qué operaciones con qué objetos. Se la considera un mecanismo obligatorio, aunque en caso de haber combinaciones de roles/objeto sin permiso asociado (una vez se apliquen los mecanismos de propagación detallados más adelante), puede asumirse el valor "Navegación" para ellas, aunque el modelo no obliga a ello (v.g. puede asumirse que la ausencia de un permiso significa no acceso). Es muy importante resaltar que el hecho de tener categorías de seguridad ordenadas provoca la existencia de permisos que "engloban" a otros permisos, en el sentido de que, además de permitir hacer lo mismo que ellos, añaden más privilegios. La función de acreditación equivale a la relación permiso-rol de RBAC.

El segundo mecanismo es la **`función de confidencialidad`**, que relaciona objetos con sujetos mediante la semántica de no acceso. Es decir, permite definir en cada objeto una lista de control de acceso negativo (nACL) en la que se indiquen los sujetos que no tienen ningún tipo de interacción con el objeto. Este mecanismo de permisos negativos, basado en una matriz de no accesos, proporciona confidencialidad, y lógicamente, tiene precedencia con respecto a los permisos positivos. A pesar de fundamentarse en modelos DAC, concretamente, en modelos basados en autoridad, la administración de este mecanismo puede dejarse en el lado del administrador de seguridad o bien a los usuarios (v.g. el dueño del objeto). Este extremo se sale del ámbito del modelo, ya que MARAH no contempla por el momento aspectos administrativos.

El tercer mecanismo es la **`clasificación de objetos`**, que asocia a un objeto una categoría de seguridad, al estilo de los modelos MAC. Esta categoría indica el tipo de manipulación más permisivo que es tolerado por el objeto. De esta forma, operaciones etiquetadas con categorías superiores no podrán ser aplicadas al objeto, sin importar los permisos del sujeto que trata de ejecutarlas. Hay varias diferencias con el concepto MAC. La clasificación de objetos no es un mecanismo obligatorio, ya que puede asumirse un valor permisivo por defecto. Además, si bien se tendría una rejilla de objetos por un lado, por otro no serían los sujetos los que estarían etiquetados, sino que son las operaciones las que poseen un nivel de seguridad que les permite ser o no ejecutadas sobre un objeto.

La clasificación de objetos permite, al igual que la función de confidencialidad, restringir la aplicación de los permisos concedidos en la función de acreditación, ya que a pesar de éstos, si el objeto tiene una etiqueta más restrictiva que la de la operación a ejecutar, ésta no se llevará a cabo. Un buen ejemplo de uso es durante la actualización de un sistema. Bastaría asignar la categoría de Navegación para evitar cualquier modificación sobre el objeto a actualizar.

Propagación de autorizaciones

Como ya se ha comentado en la sección 7.2.4, “Control de acceso basado en roles”, el establecimiento de jerarquías de roles permite recoger mejor la estructura organizativa, y proporciona una forma natural y eficaz de propagar permisos que simplifica tremendamente la gestión. MARAH establece unas reglas que gobiernan la forma de propagar valores de cada uno de los tres mecanismos de autorización. Además de la jerarquía de roles, se hace uso de la estructura de objetos para propagar dichos valores.

El primer mecanismo de propagación se aplica sobre valores de la función de confidencialidad. La regla es sencilla: un objeto añade a su nACL las nACL de todos sus objetos padre, independientemente del tipo de relación. En otras palabras, los capítulos de un libro denegarán el acceso a aquellos sujetos que no tengan acceso al nodo libro que los agrega, sin necesidad de establecer repetidamente esos permisos negativos.

El segundo mecanismo de propagación se aplica a valores de acreditaciones. El estándar de RBAC realiza la herencia simplemente añadiendo los permisos de los roles *junior* a los permisos del rol *senior*, sin entrar en detalles. En MARAH, es suficiente con que un rol tenga un único permiso por cada objeto para conocer todas las posibles interacciones. Esto es así porque las categorías de seguridad están totalmente ordenadas. En otras palabras, tener el permiso de edición de un objeto permite también personalizarlo, de manera que no es preciso incluir el permiso de personalización.

Así, se definen reglas de propagación de permisos por la estructura de roles, que permiten obtener un permiso para cada objeto del sistema por cada rol. El objetivo es, priorizando los permisos negativos y las generalizaciones (ya que representan una relación semánticamente mas fuerte que las agregaciones), diseñar un conjunto de reglas que simplifiquen la administración, proporcionando propagaciones consistentes con la semántica de las relaciones. En concreto se definen las siguientes reglas:

- **R1. Propagación directa**: cada rol hereda los permisos concedidos a su rol padre, teniendo prioridad los permisos negativos sobre los positivos. Esta regla aplica, por tanto, a la relación de confidencialidad y a la de acreditación.
- **R2. Sobreescritura de autorizaciones**: cualquier regla de propagación se anula si el rol tiene asignado explícitamente un permiso para el objeto.
- **R3. Propagación en relaciones anidadas**: si hay una generalización anidada (un rol A generaliza a otro B, que hace lo propio con otro C), el rol hijo asume los permisos que aplican al rol más especializado, es decir, a su predecesor inmediato en la jerarquía (en el ejemplo, C heredaría de B).
- **R4. Propagación en relaciones paralelas**: Si un rol hijo es generalizado por varios roles simultáneamente, éste asume la autorización más permisiva. Esto permite "recoger" el mejor de los permisos disponibles, que, por definición, englobará a los demás. Esta regla tiene una excepción, ya que los permisos negativos tienen precedencia en la herencia, de forma que si algún padre tuviera acceso denegado para el objeto en cuestión, el hijo tomaría ese valor.
- **R5. Asignación directa de permisos**: Si un rol forma parte de un equipo, y no tiene autorización (ya sea asignada directamente o heredada), tomará los permisos asignados al equipo. Si el rol forma parte de varios equipos, tomaría la autorización más permisiva.

Este conjunto de reglas permite propagar los permisos de los roles *junior* a los *senior* sin necesidad de indicar explícitamente los permisos en todos los roles de la jerarquía. Se aplicarían una vez que las nACL se hayan propagado por la estructura de objetos, de forma que los permisos negativos se calculan primero. Una vez conocidos éstos, participan en la propagación por la estructura de sujetos de la misma forma que lo hacen los permisos positivos, si bien en caso de participar en una herencia múltiple (R4), se propagarán en lugar de los positivos.

Las reglas anteriores se aplican para cada rol y cada dominio. Como es lógico, no se especifican todos los permisos de cada rol con cada objeto, por tanto, de la misma forma que se usan roles *junior*, se usan dominios para expresar permisos. Así, una vez se han propagado los permisos positivos por la estructura de roles, no tiene por qué existir un permiso por cada rol para cada objeto. En este caso, los permisos que faltan pueden calcularse empleando la jerarquía de objetos, de la siguiente forma: si un rol no tiene definido el permiso para un objeto *o*, puede buscar permisos para objetos en cuyo dominio se encuentre *o*. Así, se aplicarían reglas similares a las expuestas para ir "ascendiendo" en la jerarquía de objetos con el fin de identificar los permisos de los diferentes dominios y propagar el más permisivo para cada objeto, priorizando de nuevo los permisos negativos sobre los positivos, las generalizaciones entre objetos sobre las agregaciones, y la propagación del permiso más "cercano" al objeto. Una vez recorrida la estructura de objetos, los roles que todavía no tengan permisos para ciertos objetos pueden tomar los permisos definidos en los roles padre para los objetos

padre, lo que supone buscarlos simultáneamente por ambas jerarquías, tomando como referencia la jerarquía de roles y recorriendo los dominios correspondientes por cada rol padre.

La combinación de los dos mecanismos anteriores da lugar a lo que MARAH denomina **`regla de autorización`**. Esta regla indica, para cada sujeto, las habilidades de manipulación que tiene sobre un objeto. Así, recoge en una única función tanto los permisos positivos como negativos.

Por último, el tercer mecanismo propaga valores de la clasificación de objetos. Estos valores se heredan de acuerdo a la jerarquía de objetos, usando los mismos principios: se heredaría sólo si no se define explícitamente una categoría, se tomaría la del padre más próximo, se priorizarían las relaciones de generalización y se tomaría la categoría más permisiva de entre las disponibles en caso de haber herencia múltiple.

7.5.5 Decisión de control de acceso

Una vez vistos los mecanismos de autorización, queda por determinar cómo se toma una decisión de control de acceso, en otras palabras, cómo funciona el punto de decisión de la política. Al haber tres mecanismos para expresar autorizaciones, la decisión de acceso se basa en tres comprobaciones. Así, MARAH define la **`función de transición`** como una función lógica que indica si una petición de acceso puede ser llevada a cabo.

En su forma más simple, una petición de acceso involucra a un usuario, una operación y un objeto, aunque puede ser fácilmente generalizada para soportar varias operaciones sobre varios objetos. Por ejemplo, acceder a un nodo realmente implica leer el nodo, todos sus contenidos, todas sus anclas y todos sus enlaces.

La función de transición comprueba tres cosas:

1. **La operación puede ser ejecutada en el objeto**: para cumplir este requisito, el valor de la clasificación de la operación a ejecutar debe ser menos permisivo (o igual) que el valor de la clasificación del objeto al que se aplica
2. **No existe permiso negativo**: la regla de autorización no debe devolver un permiso procedente de una nACL
3. **El sujeto es capaz de efectuar la operación**: la regla de autorización debe devolver un valor de acreditación igual o más permisivo que la clasificación de la operación. En otras palabras, debe haber un permiso positivo que denote privilegios suficientes.

De cara a la implantación del modelo en un sistema, es importante tener en cuenta que es preciso ejecutar esta función en todos los puntos de cumplimiento de la política. Por ello, en la especificación de Labyrinth, toda operación incluye implícitamente una llamada a la función de transición antes de ser ejecutada.

7.6 Ejemplo de diseño de la seguridad en un sistema de información web

En esta sección se describe la integración del modelo MARAH con el método de desarrollo Ariadne (ADM, *Ariadne Development Method*) descrito en el capítulo 2. Para ello se prestará especial atención a las fases del método relacionadas con la especificación de la política de acceso, tomando como ejemplo de desarrollo el sistema ARCE.

ARCE es un proyecto iberoamericano en el que los 21 países que forman la Asociación Iberoamericana de Organismos de Defensa y Protección Civil cooperan para crear una plataforma web que permita mejorar la asistencia en situaciones de emergencia [43]. El sistema ARCE ofrece un canal de comunicaciones actualizado, fiable, rápido y flexible que permite compartir información entre los miembros de la asociación para coordinar una respuesta integrada y eficiente cuando un desastre ocurre.

Como se ha visto en el capítulo 2, ADM tiene tres fases: diseño conceptual, diseño detallado y evaluación. En el diseño conceptual de ARCE se definen los conceptos fundamentales de la aplicación a un nivel abstracto, mientras que en el detallado se especificarían las instancias, bien de forma declarativa (v.g. ciertos objetos que existen por cada país de la asociación), bien de forma procedural (v.g. el proceso de creación de una emergencia). En la evaluación se realizan comprobaciones empíricas sobre la implementación, aunque también se evalúan modelos y prototipos.

7.6.1 Diseño conceptual

Del diseño conceptual se hará hincapié en cuatro fases, en las que se definen los cinco productos relacionados con la seguridad: el diagrama estructural, las especificaciones funcionales, diagrama de usuarios, reglas de autorización y el catálogo de categorizaciones.

La primera actividad es la **definición de la estructura lógica**, en la que se crea el **diagrama estructural**, donde se establecen los tipos de entidad que forman parte de la aplicación, y sus relaciones de agregación y generalización. Este diagrama recoge el modelado de objetos de MARAH. La Figura 7.4 muestra la parte del diagrama estructural relacionada con las emergencias en ARCE. Los nodos simples tienen un sólo borde, mientras que los compuestos tienen doble borde.

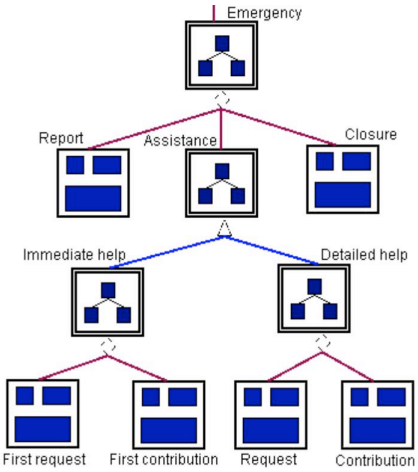


Fig. 7.4: Extracto del modelado de objetos en ARCE

Puede verse cómo el nodo superior, que representa las emergencias, se compone de tres nodos: el informe inicial de la emergencia, un nodo compuesto que representa la ayuda relacionada con la emergencia, y un informe de cierre de emergencia. A su vez, el nodo de ayuda se especializa en ayuda inmediata, aquella que se produce en las 48 primeras horas tras la situación de emergencia, y en ayuda detallada. Por último, cada nodo de ayuda se especializa en dos nodos, uno que representa las peticiones y otro que representa las contribuciones.

La siguiente actividad es el **estudio de las funciones del sistema**, en la que se crean el **diagrama de navegación** y las **especificaciones funcionales**. El primero especifica las rutas y herramientas de navegación, y las segundas reflejan servicios no relacionados con la navegación. La navegación, si bien puede verse afectada por el control de acceso, no está relacionada con la definición de la política de accesos. Las especificaciones funcionales recogen funciones pertenecientes al dominio de aplicación del sistema, que podrán descomponerse en subfunciones. Cada especificación funcional incluye el identificador de la función, su nombre, una descripción, el tipo, subfunciones, caso de existir, y eventos relacionados. Todas las funciones identificadas se enumeran en el catálogo de funciones.

La Figura 7.5 muestra las especificaciones de la función "Gestionar emergencia", en la que se puede ver cómo esta función se subdivide en funciones como "Abrir emergencia", "Actualizar emergencia" o "Solicitar ayuda". En el diseño detallado se especifica con todo detalle la estructura de cada función.

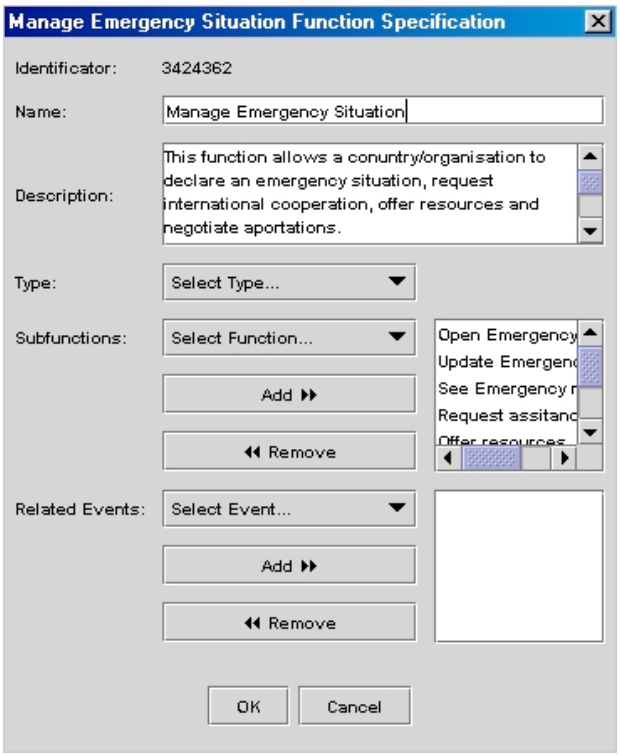


Fig. 7.5: Especificaciones funcionales para la función ARCE Gestión de Emergencia

La siguiente actividad es la **especificación de entidades**, que no tiene relación con la especificación de la política de acceso, ya que define los diagramas internos (en los que se especifica la ubicación espacio/temporal de los contenidos en los nodos) y los catálogos de atributos y eventos.

La siguiente actividad es el **modelado de usuarios**, en la que se crea el **diagrama de usuarios**. Este diagrama recoge la estructura organizativa en la que se implantará el sistema, en términos de roles y equipos, nunca tratando con usuarios concretos. Recoge exactamente el modelado de sujetos de MARAH. La Figura 7.6 muestra el diagrama de usuarios de ARCE, que modela en un único producto todas las estructuras de las organizaciones participantes en la asociación. Los roles se representan mediante cajas con una cara, mientras que los equipos son cajas con dos caras.

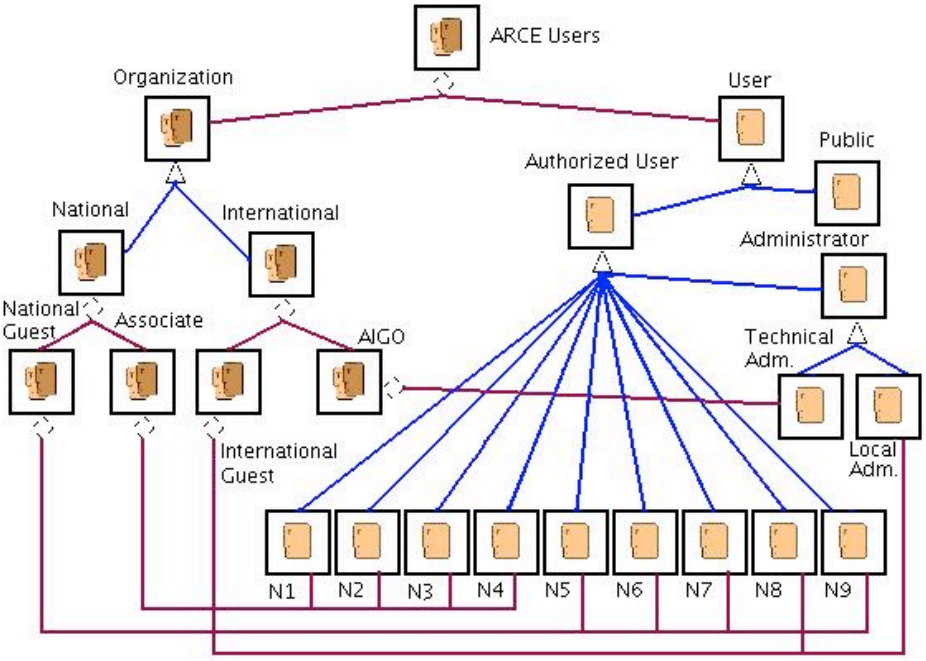


Fig. 7.6: Diagrama de usuarios para ARCE

Uno de los roles que merece especial mención es el N4. N4 es una especialización de Usuario Autorizado (rol que recoge varios permisos por defecto), y a la vez forma parte del equipo de organismos asociados, que, como puede observarse, es un subequipo del equipo que representa organismos nacionales. De esta forma, permisos concedidos al Usuario Autorizado serán propagados a N4 antes que los concedidos al equipo Asociado. Por último, el equipo Organización no está compuesto de roles, sino que es una generalización de dos equipos diferentes, Nacional e Internacional.

La última actividad del diseño conceptual es la definición de la política de seguridad. Obviamente, a este nivel del diseño el esfuerzo se centra en la política de accesos, especificada mediante dos productos: el **catálogo de categorizaciones** y **las reglas de autorización**. El catálogo de categorizaciones recoge la función de clasificación de objetos de MARAH, es decir, permite asociar a cada objeto la etiqueta de seguridad que indica el tipo de manipulación más permisiva que se permitirá sobre el objeto. Para ARCE, se asigna por omisión la categoría de Edición ya que la mayor parte de los objetos requieren ser modificados frecuentemente. En todo caso, cuando se cierra una emergencia, todos los objetos relacionados con ésta cambian su categoría a Navegación, para así no poder ser modificados por ningún usuario, independientemente del rol o roles asignados al mismo.

Las reglas de autorización asignan a cada rol o equipo las funciones que van a poder realizar en el sistema, de aquellas identificadas en las especificaciones funcionales. Cuando un rol tiene autorización para realizar una función, automáticamente la tiene

para todas sus subfunciones. La Figura 7.7 muestra las reglas de autorización para ARCE.

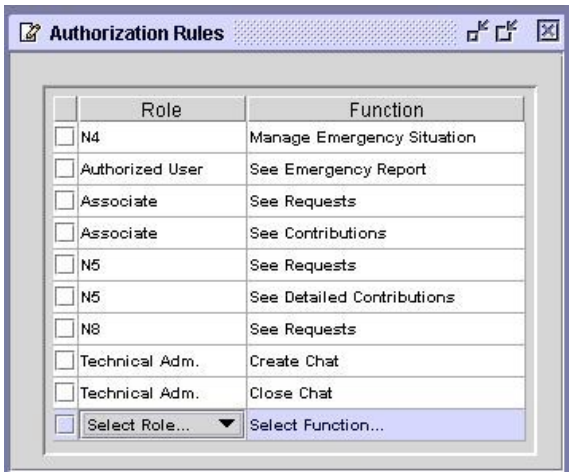


Fig. 7.7: Reglas de autorización del sistema ARCE

Puede observarse que la asignación es muy fácil de realizar, y se corresponde con lo que se definió en MARAH como nivel de aplicación. Simplemente se listan los roles definidos en el diagrama de usuarios y las funciones, pudiendo hacerse emparejamientos según la política de accesos. Por ejemplo, el rol N4 está autorizado para gestionar las emergencias, y por tanto va a poder solicitar ayuda, ofrecerla, etc. Por otro lado, el rol N8, que representa miembros de organizaciones internacionales invitadas a participar (v.g. Naciones Unidas) puede solamente visualizar las peticiones de ayuda. Finalmente, puede observarse que hay autorizaciones definidas para equipos, que se propagarán por la estructura de sujetos una vez se hayan transformado en permisos.

Lógicamente, estas reglas no tienen representación directa en MARAH, ya que este modelo trabaja a un nivel independiente del dominio. En el diseño detallado se verá cómo estas autorizaciones tienen reflejo en forma de permisos MARAH.

7.6.2 Diseño detallado

En esta fase, centrada en un nivel de abstracción menor que la fase anterior, se especifican las entidades concretas que se van a crear en el sistema final, bien de forma declarativa, bien de forma procedural.

La primera actividad es la **identificación de instancias**, que da lugar al **diagrama de nodos instanciados** y al **diagrama de usuarios instanciado**. Cada instancia tiene un atributo especial, denominado descriptor de instancia, cuyo valor la identifica unívocamente. En ARCE todas las instancias se generan en virtud del nombre del país

a la que pertenecen. De hecho, se usa este nombre como descriptor de instancia, de forma que el nodo "Emergencia" del diseño conceptual pasa a tener varias instancias como "Emergencia.Argentina" o "Emergencia.Guatemala". De forma análoga se procede con el diagrama de usuarios, que ahora se instancia para cada país de manera que se tienen instancias como "N4.Argentina" o "N4.Guatemala". Este descriptor es clave en el cálculo de las reglas finales de acceso, como se comenta más adelante.

La siguiente actividad es la **especificación de funciones**, en la que se crean la **especificación de las estructuras de acceso** y la **especificación detallada de funciones**. En este último producto se describe la descomposición funcional iniciada en el conceptual. Las funciones compuestas se dividen en subfunciones, hasta llegar a funciones simples, las cuales se descomponen en un cierto número de operaciones atómicas del modelo Labyrinth, que operan sobre tipos de objetos (nodos, contenidos, anclas, enlaces, atributos y eventos). Además, pueden indicarse estructuras de control para especificar cuántas veces se ejecuta una determinada operación. La Tabla 7.1 muestra la descomposición funcional de la función "Crear informe de emergencia", que se correspondería con la función "Open Emergency" de la Figura 7.5.

Tabla 7.1: Especificación detallada de funciones para la función ARCE Crear Informe de Emergencia

```
CreateNode(Informe)
# Los contenidos del informe se crean automáticamente
CreateContent(Tipo de Emergencia)
CreateContent(Lugar de la Emergencia)
CreateContent(Daños)
...
# Los contenidos se sitúan automáticamente en el nodo
placeContentNode(Tipo de Emergencia, Informe)
placeContentNode(Lugar de la Emergencia, Informe)
...
# Modificaciones del usuario en el informe
1{EditContent(user, Lugar de la Emergencia)}N
1{EditContent(user, Tipo de Emergencia)}N
0{EditContent(user, Daños)}N
...
```

Puede observarse cómo una función simple se ha descompuesto en un conjunto de operaciones sobre elementos abstractos del modelo, tendiéndose así un puente entre conceptos propios de la aplicación y operaciones del nivel de hipermedia. La notación `0{<operaciones>}N` indica que el conjunto de operaciones encerrado entre llaves se ejecutará de 0 a N veces en cada ejecución de la función simple.

La siguiente actividad es la *especificación de instancias*, que da lugar a los **diagramas internos detallados**, la **tabla de accesos** y **la asignación de usuarios**. El primer producto detalla completamente la composición de cada instancia de nodo y contenido, incluyendo referencias a la ubicación física de los recursos (URIs).

La tabla de accesos almacena los derechos de acceso de cada instancia de sujeto (roles/equipos) para cada instancia de objeto (véase Tabla 7.2). Las celdas de esta tabla se calculan automáticamente a partir de las especificaciones de funciones y las reglas de autorización, de la forma siguiente: a todo aquel sujeto autorizado para ejecutar una función se le asignan los privilegios mínimos para ejecutar cada una de las operaciones atómicas que forman parte de la función. Por ejemplo, si el rol N4 puede ejecutar la función "Crear informe de emergencia" (tabla 7.1), requerirá que se le asigne el permiso de Edición para el nodo Informe, así como para todos los contenidos que forman parte del nodo (Tipo de Emergencia, Lugar, Daños, ...). Una vez asignadas estas acreditaciones, explicadas con más detalle en [44], se pondrían en marcha los mecanismos de herencia definidos en MARAH para propagar los permisos por todos los roles. De forma similar se procedería con el resto de reglas de autorización definidas en la Figura 7.7.

Tabla 7.2: Fragmento de la Matriz de Accesos para el sistema ARCE

Rol	Objeto	Manipulación permitida
N4	Informe	Edición
N4	Tipo de Emergencia	Edición
N4	Lugar de Emergencia	Edición
N4	Daños	Edición

Gracias a las especificaciones funcionales detalladas y a los mecanismos de herencia, puede rellenarse completamente la Tabla 7.2. Pero estas reglas de acceso siguen siendo conceptuales, ya que se refieren a nodos del diseño conceptual, y no a instancias. Para generar reglas instanciadas de acceso, hay que recurrir al descriptor de instancia, de forma que se relacionan aquellas instancias de sujetos y objetos cuyo descriptor coincida. Así, N4.Argentina tendrá asignado el permiso de Edición para Emergencia.Argentina, mientras que puede asignársele permiso de Navegación para el resto de situaciones de emergencia. Así, se pueden definir permisos dependientes del contexto. Estas autorizaciones se corresponden exactamente con el concepto de permiso en MARAH.

Por último, la asignación de usuarios relaciona usuarios concretos del sistema final con los roles que van a poder desempeñar. Este modelo se corresponde completamente con la asignación de usuarios en RBAC. Si se utiliza un enfoque RBAC puro como es el caso de ARCE, ya se dispone de toda la información necesaria para conocer los derechos de acceso de cada usuario en el sistema. Si se utiliza **`control de acceso basado en credenciales`** (CBAC, *credential-based access control*), la definición de cada rol, excluyendo equipos, contendría el conjunto de **`tipos de credencial`** que deben ser satisfechos para poder desempeñar el rol. Un tipo de credencial obliga a acreditar un determinado conjunto de valores de atributos para que los usuarios puedan poseer una credencial de ese tipo. Estos atributos y valores se indican mediante expresiones regulares sobre los atributos definidos en el diagrama de usuarios y el catálogo de atributos. De esta forma, en tiempo de ejecución, los usuarios que posean las credenciales requeridas por un rol serán automáticamente asignados a él, disfrutando desde ese momento de los derechos de acceso otorgados a ese rol.

7.6.3 Evaluación

La evaluación se centra en mejorar el sistema a través de un proceso de evaluación continua, que puede aplicarse a diseños o a prototipos. En ARCE, los modelos de diseño son evaluados con expertos de la asociación, lo cual permite discutir los requisitos de acceso desde el principio del proyecto e incluso detectar duplicidades o asignaciones erróneas en las diferentes organizaciones.

Por otro lado, periódicamente se realizan evaluaciones empíricas de prototipos [43], proponiendo un ejercicio de simulación en el que un miembro de la asociación crea una emergencia y el resto de los asociados simulan el proceso de contribución para mitigar sus efectos. Así, las reglas de acceso y la estructura de sujetos se van depurando iterativamente con el objeto de recoger mejor las características y requisitos de los usuarios de ARCE.

7.7 Conclusiones y trabajo futuro

Los aspectos de seguridad cobran cada vez más relevancia en el desarrollo de software, más si cabe al considerar aplicaciones que se ejecutan en un contexto abierto como es Internet. Por ello estos aspectos no pueden ser relegados al momento de implementar o implantar el sistema. La integración del diseño de la seguridad con el diseño del resto del sistema es crucial.

Bajo esta premisa, este capítulo ha recorrido los aspectos más relevantes de la seguridad que pueden ser considerados durante todo el desarrollo, centrándose en el control de accesos, teniendo en cuenta la necesaria formalización de los modelos sobre los que descansan, y su inclusión en marcos metodológicos a fin de proporcionar una guía durante el ciclo de desarrollo.

Se ha presentado el modelo de acceso MARAH, que, fundamentado en el principio de abstracción, ofrece mecanismos para expresar de forma sencilla y precisa la política de accesos a un sistema hipermedia. Esta sencillez se materializa en la filosofía basada en roles y equipos, a lo que se añaden mecanismos adicionales para simplificar más la labor del administrador y del diseñador de la política, como son la jerarquización de objetos, la clasificación de éstos de acuerdo a las operaciones que pueden soportar o el uso de listas de control de acceso negativas.

De cara al trabajo futuro, pueden esbozarse varias líneas de mejora. Referente al método, hay que estudiar con profundidad las implicaciones que puede suponer la definición de permisos negativos en el diseño detallado, una vez se han calculado automáticamente los permisos positivos en virtud de la descomposición funcional y las asignaciones de roles con funciones del dominio de la aplicación, con vistas a garantizar que dichas funciones continúan estando disponibles para los usuarios autorizados.

Respecto al modelo MARAH como tal, existen varias líneas previstas que lo extienden, entre las que se encuentran las siguientes:

- Abarcar el tiempo de diseño, de forma que el control de acceso se aplique también a operaciones relacionadas con la modificación de la estructura del hiperdocumento.
- Añadir capacidades administrativas: la gestión de roles, equipos, permisos y usuarios estaría regulada de acuerdo a una política RBAC.
- Delegación de permisos: permitiría a determinados usuarios delegar uno o más permisos a otros durante un tiempo determinado. Incluye aspectos como la delegación en cascada, que permite o no la delegación de un permiso delegado; la validez del permiso delegado, en tanto a la duración o número de veces que puede hacerse uso de él; o la revocación de permisos.
- Integración con operaciones dependientes del dominio. Esta extensión pretende facilitar el paso de operaciones de alto nivel definidas en el nivel de aplicación a operaciones MARAH a través de un proceso de descomposición funcional, que permitiría especificar la política de seguridad al nivel de aplicación. Se espera que la retroalimentación proporcionada por el método Ariadne permita incluir estos formalismos en el modelo.
- Inclusión de más restricciones: sin duda, este es el aspecto más importante del modelo. Existen multitud de restricciones que pueden añadirse y que aportan gran riqueza y potencia expresiva al modelo. Algunos ejemplos son:
  - Separación de derechos: además de expresar roles excluyentes, pueden especificarse permisos excluyentes y objetos excluyentes, tanto estática como dinámicamente. Asimismo, la separación de derechos histórica prevendría a un usuario de realizar una acción en función de las acciones anteriores. Algunas características interesantes se describen en [11].
  - Cardinalidad: permite limitar el número de usuarios asignados a un rol, tanto estática como dinámicamente. Es preciso definir la semántica de la cardinalidad para roles que generalizan a otros y para equipos.
  - Restricciones temporales: permiten expresar intervalos (periódicos o no) en virtud de los cuales se puede habilitar/deshabilitar un rol, un permiso o una asignación permiso-rol. Añaden riqueza expresiva al modelo ya que permiten recoger prácticas organizativas relacionadas con horarios o plazos.
  - Restricciones dependientes del contexto: permiten limitar la aplicación de un permiso al cumplimiento de una serie de expresiones formuladas mediante parámetros cuyo valor se determina en tiempo de ejecución del sistema. Son de especial utilidad en entornos abiertos, donde el contexto de acceso puede ser muy variado y difícilmente representable únicamente mediante roles.

Agradecimientos

Este trabajo ha sido realizado en el marco del proyecto "ARCE++: Métodos avanzados de acceso a un sistema de información para la cooperación internacional en

situaciones de desastre" financiado por la Dirección General de Investigación del Ministerio de Educación y Ciencia con la referencia TSI2004-03394.

Referencias

[1] Brose, G., Koch, M. And Löhr, K.P. (2001). Integrating Security Policy Design into the Software Development Process. Technical Report B–01–06. Freie Universität Berlin, Nov. 13.

[2] Devanbu, P.T. and Stubblebine, S. (2000). Software engineering for security: a roadmap. The Future of Software Engineering. Finkelstein, A. ed. ACM Press.

[3] eXtensible Access Control Markup Language (XACML) v1.0. (2003) http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf

[4] Osborn, S., Sandhu, R. and Munawer, Q.: Configuring role-based access control to enforce mandatory and discretionary access control policies. AC Transactions on Information and System Security 3(2). (2000)

[5] Bell, D.E. and LaPadula L.J.: Secure Computer Systems: Mathematical Foundations and Model. Mitr Corp. Report No. M74-244. (1975)

[6] Harrison, M.A, Ruzzo W.L and Ullman, JD.: Protection in Operating Systems. Communications of the ACM (19)8. (1976)

[7] Saltzer, J.H. and Schroeder, M.D.: The Protection of Information in Computer Systems. Communications of the ACM 17(7). (1974)

[8] Aedo, I., Díaz, P. and Montero, S.: A methodological approach for hypermedia security modelling. Information and Software Technology, 45(5). 229–239. 2003

[9] ANSI INCITS 359-2004, American National Standard for Information Technology; Role Based Access Control. (2004)

[10] Ferraiolo, D. , Cugini, J and Kuhn, R.: Role-based access control (RBAC): Features and motivations. In Proceedings of Annual Computer Security Applications Conference, IEEE Computer Society Press (1995).

[11] Crampton, J.: Specifying and Enforcing Constraints in Role-Based Access Control. Proceedings of SACMAT 2003, Como, Italy, 43-50.

[12] Joshi, J.B.D., Bertino, E., Latif, U., Ghafoor, A.: Generalized Temporal
Role Based Access Control Model (GTRBAC) (Part I) - Specification and
Modeling. (2001) Submitted to IEEE Transaction on Knowledge and Data Engineering. Available as CERIAS technical report at:
https://www.cerias.purdue.edu/infosec/bibtex_archive/archive/2001-47.pdf

[13] Neumann, G and Strembeck, M.: An Approach to Engineer and Enforce Context Constraints in an RBAC Environment. Proceedings of SACMAT 2003, Como, Italy, 65-79.

[14] Campbell, B. and Goodman, J. M.: HAM: A general purpose hypertext abstract machine' Communications of the ACM 31 (7), 856-861. (1988).

[15] Stotts P. D. and Furuta R.: Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. ACM Transactions on Office Information Systems, 7(1). 3-29. (1989).

[16] Thuraisingham, B.: Multilevel security for information retrieval systems-II. Information and Management, 28, 49-61. (1995).

[17] Bertino, E. and Smarati, P.: Research Issues in Authorization Models for Hypertext Systems. Proc. of the 1995 ACM SIGSAC New Security Paradigms Workshop, La Jolla, CA, 22-27. (1995).

[18] Samarati, P., Bertino, E. and Jajodia, S.: An Authorization Model for a Distributed Hypertext System. IEEE Trans. on Knowledge and Data Engineering, 8 (4), 555-562. (1996).
[19] Bertino, E. Pagani, E. Rossi, G.P. and Samarati, P.: Protecting Information on the Web. Communications of the ACM , 43(11es), 189-199. (2000)
[20] Joshi, J. Aref, W. Ghafoor, A. and Spafford, E.: Security models for web-based applications, Communications of the ACM, 44(2), 38-44. (2001)
[21] Ferraiolo, D.F., Barkley, J.F. and Kuhn, D.R: A role-based access control model and reference implementation within a corporate intranet. ACM Trans. on Information and Systems Security, 2(1), 34-64. (1999).
[22] Wang, W.: Team-and-role-based organizational context and access control for coooperative hypermedia environments. Proc. of Hypertext'99, Darmstad, 37-46. (1999)
[23] W3 Consortium: Extensible Markup Language (XML) 1.1. (2004) http://www.w3.org/TR/2004/REC-xml11-20040204/
[24] Damiani, E., De Capitani di Vimercati, S., Paraboschi S. and Samarati, P.: A Fine-Grained Access Control System for XML Documents. ACM Trans. on Information and System Security, 5(2). 169-202. (2002)
[25] Bertino, E. and Ferrari, E.: Secure and selective dissemination of XML documents. ACM Trans. on Information and System Security, 5(3). 290-331. (2002)
[26] W3 Consortium. XML Path Language (XPath). (1999) http://www.w3.org/TR/xpath
[27] Zhang, X., Park, J., and Sandhu, R.: Schema Based XML Security: RBAC Approach. 17th IFIP 11.3 Working Conference on Data and Application Security. Estes Park, Colorado, USA. (2003)
[28] W3 Consortium. XML Schema Part 0: Primer Second Edition. (2004) http://www.w3.org/TR/xmlschema-0
[29] Lim, C., Park, S., and Son, S.: Access Control of XML Documents Considering Update Operations. ACM Workshop on XML Security, Fairfax VA, USA (2003)
[30] Technical Overview of the OASIS Security Assertion Markup Language (SAML) v1.1. (2004) http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf
[31] W3Consosrtium: Web Services Architecture. (2004) http://www.w3.org/TR/ws-arch/
[32] Bhatti, R., Bertino, E., Ghafoor, A. and Joshi, J.B.D.:XML-based specification for web services document security. IEEE Computer, 37(4), 41-49. (2004)
[33] Security in a Web Services World: A Proposed Architecture and Roadmap. (2002) http://www-106.ibm.com/developerworks/library/ws-secmap/
[34] Web Services Security (WS-Security). (2002) http://www-106.ibm.com/developerworks/library/ws-secure/
[35] Web Services Policy Framework (WS-Policy). (2004) http://www-128.ibm.com/developerworks/library/specification/ws-polfram/index.html
[36] Web Services Trust Language (WS-Trust). (2004) http://www-128.ibm.com/developerworks/library/specification/ws-trust/index.html
[37] Kienzle D.M. and Elder MC.: Security Patterns for Web Application Development. DARPA Technical Report. (2002) http://www.scrypt.net/~celer/securitypatterns/final%20report.pdf
[38] Fernandez, E: A pattern language for security models. (2004) http://polaris.cse.fau.edu/~ed/SecModelsV3.pdf
[39] Kodituwakku, S., Bertok. P, and Zhao, L. APLRAC: A Pattern Language for Designing and Implementing Role-Based Acess Control. In Proc. The Second Asian-Pacific Pattern Languages of Programming Conference, Monticello, IL, (KoalaPLoP'2001) 2001.

[40] Kandala, S. and Sandhu, R.: Secure role-based workflow models. Proceedings of the fifteenth annual working conference on Database and application security. Niagara, Ontario, Canada, 45-58 (2001)
[41] Díaz, P., Aedo I. and Panetsos, F.: Labyrinth, an abstract model for hypermedia applications. Description of its static components. Information Systems, 22(8), 447-464. (1997)
[42] Díaz, P., Aedo, I. and Panetsos, F.: Modelling the dynamic behaviour of hypermedia applications. IEEE Trans. on Software Engineering, 27(6), 550–572. (2001)
[43] Aedo, I., Diaz, P., Fernandez, C. and de Castro, J.: Supporting Efficient Multinational Disaster Response through a Web-Based System. Proceedings of First International Conference of Electronic Government. Aix-en-Provence, France, 215-222. (2002)
[44] Sanz, D., Aedo, I. and Diaz, P.: Using Web and Role-Based Access Control for Efficient Multinational Disaster Response. Proceedings of IADIS e-Society 2004 conference, Avila, Spain, 63-70. (2004)

8 CALIDAD EN EL DESARROLLO DE APLICACIONES WEB: MODELOS, MÉTODOS DE EVALUACIÓN Y MÉTRICAS DE CALIDAD

María Dolores Lozano, Francisco Montero, Fco. Javier García, Pascual González

Laboratorio de Interacción con el Usuario e Ingeniería del Software
Dpto. de Informática
Universidad de Castilla-La Mancha

8.1 Introducción

En los últimos años hemos presenciado el rápido desarrollo de Internet y la sofisticación de la tecnología informática asociada, lo que ha permitido su integración en casi todos los sectores de la sociedad, como hogares, educación, comercios, salud y gobierno y, con ello, la creciente complejidad de las interfaces de usuario de estos sistemas. Todos estos avances hacen que cada vez sea más importante y difícil la tarea de garantizar la calidad de estos sistemas que, de un modo u otro, se convierten en casi imprescindibles para la sociedad actual.

Las aplicaciones software centradas en la web son cada vez más complejas. Esto es debido fundamentalmente a que muchas aplicaciones de gestión tradicional han migrado a entorno web o incluido una nueva visión del sistema ligada a este entorno. Por otra parte, la facilidad de desarrollar aplicaciones en las que la ubicación física del usuario no es importante, ha abierto el camino al desarrollo de otro tipo de sistemas, como los entornos colaborativos, los sistemas de tele-enseñanza, etc. De la misma manera, el número de usuarios que utilizan estas tecnologías también ha aumentado. Es necesario, por tanto, acompañar este rápido crecimiento con modelos de proceso y metodologías de evaluación que permitan mejorar la calidad de las aplicaciones Web para permitir una mejor interacción entre usuarios y estos sistemas.

Las prácticas actuales para desarrollar sitios web son *ad-hoc*, basada en el sentido común y en la experiencia de los desarrolladores, y la garantía y control de la calidad es, por lo general, un proceso a cumplimentar en el futuro. La utilización sistemática y disciplinada de métodos, modelos, y técnicas de Ingeniería de Software para el desarrollo, el mantenimiento, y la evaluación de la calidad de los sitios web ha de ser un requisito obligatorio, principalmente en los proyectos de mediana o gran escala.

La calidad del software en general ha sido, y es, uno de los temas de mayor preocupación y más estudiados en relación con el desarrollo de software. De hecho, la evaluación de sistemas software se lleva realizando desde hace más de tres décadas [9] [22], pero hoy en día la evaluación de sitios y aplicaciones web es todavía reciente y en continuo desarrollo. Esto se debe a que las aproximaciones tradicionales no consideran los cambios impuestos por las nuevas tecnologías y es necesario, por tanto, adaptar los procesos, modelos y métricas existentes para estimar proyectos web.

Además, hay que tener en cuenta que la calidad de un producto software está directamente relacionada con la calidad del proceso llevado a cabo. Por tanto, la mejora del proceso de desarrollo influye en la mejora de la calidad del producto software desarrollado finalmente. El estudio de la calidad de productos y procesos de desarrollo para la web es muy reciente y no se dispone de métodos de evaluación estandarizados para este tipo de entorno [28].

Teniendo en cuenta estos aspectos, en este capítulo se aborda el tema de la calidad del software para entornos web, incidiendo especialmente en uno de los factores de calidad más importantes en este tipo de entornos: la usabilidad. Para ello, la estructura del capítulo se organiza de la siguiente manera: En la segunda sección, se analizan diferentes métodos, técnicas y métricas para evaluar y medir la usabilidad de aplicaciones web y se introduce el concepto de modelo de calidad web. En la tercera sección, y una vez introducido dicho concepto, se realiza una revisión y comparativa de los modelos de calidad más destacados. La cuarta sección aborda el uso de métricas de usabilidad como técnica de medida y validación de la usabilidad en la web, analizando diferentes propuestas. En la quinta sección se discuten diferentes aproximaciones de integración de la calidad en el propio proceso de desarrollo, ya que como se ha comentado anteriormente, la calidad del proceso influye decisivamente en la calidad del producto final. En este sentido, se analizan algunas propuestas planteadas y se propone un marco metodológico que trata de aportar soluciones al problema de incorporar criterios de usabilidad dentro del proceso de desarrollo de aplicaciones web. Para finalizar el capítulo, la última sección recoge las conclusiones y los aspectos destacados del capítulo.

8.2 Evaluación de la calidad web: métodos, técnicas y uso de métricas de usabilidad

8.2.1 Introducción a la Calidad Web

La norma ISO 8402 [12], define la calidad como:

El conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer unas necesidades explícitas o implícitas.

Según esta definición, la calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, facilidad de mantenimiento, portabilidad, facilidad de uso, seguridad, integridad, etc. En definitiva, la calidad consta de toda una serie de factores que deberán ser medidos.

Como se puede apreciar, la evaluación de un sistema software es compleja. Esto se debe a que ésta se expresa en función de varias características, subcaracterísticas y atributos. Por ello, se han definido distintos modelos que permiten descomponer en atributos fáciles de medir los factores representativos que puedan servir para dar una visión global sobre la calidad de ese producto software.

Los primeros modelos de calidad software que surgieron son los de Mc Call [22] y Boehm [5], los cuales se centran en los factores que son claves para el producto software. En el caso del modelo de Mc Call, los factores se descomponen en criterios más específicos que se pueden medir por medio de un conjunto de métricas.

Basándose en el modelo de Mc Call, surgió el modelo ISO 9126 [11]; un modelo normalizado que permite evaluar y comparar productos sobre la misma base. En este modelo, la calidad queda definida a un alto nivel de abstracción por seis características principales: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

- **Funcionalidad**: Las funciones satisfacen las necesidades declaradas o implícitas (ISO 9126: 1991, 4.1) Adecuación; exactitud; interoperabilidad; seguridad de acceso; cumplimiento funcional.
- **Fiabilidad**: Capacidad de un sistema para mantener su nivel de rendimiento, en condiciones establecidas y durante un período de tiempo determinado. (ISO 9126: 1991, 4.2). Madurez; tolerancia a fallos; capacidad de recuperación.
- **Usabilidad**: Esfuerzo necesario para el uso y la valoración individual de tal uso por parte de un conjunto de usuarios declarado o implícito. (ISO 9126: 1991, 4.3). Capacidad para ser entendido; capacidad para ser aprendido; capacidad para ser operado.
- **Eficiencia**: Relación entre el nivel de prestaciones de un sistema y el volumen de recursos utilizados en condiciones declaradas. (ISO 9126: 1991, 4.4). Comportamiento temporal; utilización de recursos.
- **Mantenibilidad**: Esfuerzo necesario para realizar modificaciones específicas. (ISO 9126: 1991, 4.5). Capacidad para ser analizado; capacidad para ser cambiado; estabilidad; capacidad para ser probado.
- **Portabilidad** Capacidad de un sistema para ser transferido de un entorno a otro. (ISO 9126: 1991, 4.6). Adaptabilidad; capacidad de ser instalado; capacidad para reemplazar.

Estas características son muy genéricas por lo que para su evaluación se emplea un proceso de descomposición recursivo que se basa en subcaracterísticas y atributos teniendo en cuenta una meta de evaluación y un perfil de usuario dados.

Este estándar no proporciona métricas ni métodos para la medición y evaluación de dichas características. En consecuencia, no son prácticas las mediciones directas de las características de calidad definidas en este estándar.

Las definiciones ISO 9126 [11] muestran que el objetivo es encontrar las necesidades del usuario. Sin embargo, la ISO 8402 [12] deja claro que la calidad está determinada por la presencia o la ausencia de atributos, con la implicación de que estos son atributos específicos que pueden ser diseñados en el producto. ISO 8402 distingue esta vista de calidad de las medidas del "grado de excelencia" resultado de la presencia o ausencia de atributos requeridos. Todavía el objetivo de calidad desde la perspectiva de usuario es para el software exhibir excelencia en las condiciones actuales de uso.

Para resolver este problema, ISO 9126 se revisó para incluir un nuevo modelo de calidad que distingue entre 3 diferentes aproximaciones a la calidad de producto en ISO 14598. [13].

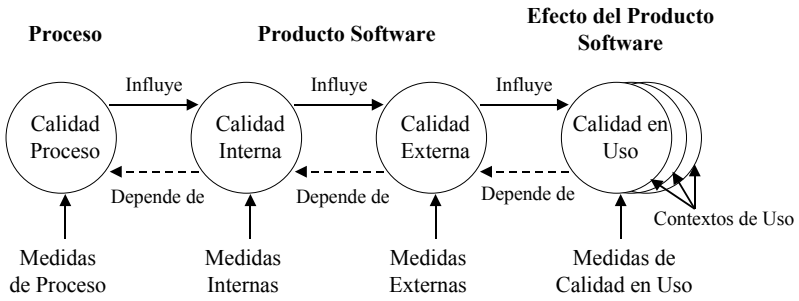


Figura 8.1 Estándar ISO 9126

- **Calidad interna:** se mide por las propiedades estáticas del código, utilizando normalmente técnicas de inspección.
- **Calidad externa:** se mide por las propiedades dinámicas del código cuando éste se ejecuta.
- **Calidad en uso:** se mide por el grado por el cual el software está realizado en función de las necesidades del usuario en el entorno de trabajo para el que fue construido.

La ISO 14598-1 [13] define la **calidad externa** como el instante en el cual un producto satisface las necesidades establecidas e implícitas cuando éste es utilizado bajo ciertas condiciones especificas. La calidad externa es el resultado de combinar el comportamiento del software y del sistema. Con ello el término calidad del producto deja de ser algo aislado, y pasa a tener en cuenta la satisfacción de unas necesidades de unos usuarios en unas circunstancias particulares.

Se puede decir por tanto que las características de calidad software en la revisión de ISO/IEC 9126 se han redefinido en términos de "capacidad del software", para permitirles ser interpretadas con una perspectiva tanto interna como externa. Las definiciones también se refieren al "uso bajo condiciones específicas" para dejar claro que *la calidad no es una característica absoluta*, sino que *depende del contexto de uso*.

La **calidad en uso** es la efectividad, productividad y satisfacción del usuario cuando soporta tareas representativas en un entorno realista de trabajo [4], definido por la ISO 14598-1 [13] como la efectividad, eficiencia y satisfacción con la cual ciertos usuarios específicos pueden alcanzar ciertas metas en entornos concretos. Ese concepto está muy próximo a la definición de usabilidad que se verá en apartados siguientes.

Las medidas externas se pueden usar para validar la calidad interna del software. La calidad en uso mide el grado de excelencia y puede ser usada para validar el grado en que el software conoce las necesidades de usuario. Los atributos internos apropiados del software son un prerrequisito para la realización del comportamiento externo requerido, y el comportamiento externo apropiado es un prerrequisito para la realización de la calidad en uso.

Hasta ahora todas estas definiciones se refieren a calidad de software en general. A continuación se define en particular qué se entiende por calidad de sistemas web.

La **calidad de sistemas web** básicamente se fundamenta bajo los mismos principios que la calidad del software en general, ya que a fin de cuentas una aplicación web es un producto software, aunque eso sí, con ciertas propiedades características de este tipo de entornos que lo hacen en cierto modo diferente de las aplicaciones tradicionales. En [26] y en el capítulo 1 de este libro, se pueden encontrar algunas de las características principales que diferencian el software tradicional del software para entornos Web. Además, la complejidad de los sistemas Web ha ido en aumento, pasando de meros repositorios de información, a aplicaciones cada vez más complejas tanto en estructura, funcionalidad, como en interfaz. Dada la complejidad y variedad de estas aplicaciones, en la literatura podemos encontrar diferentes clasificaciones [10] tales como: aplicaciones interactivas, transaccionales, flujos de trabajo, entornos de trabajo colaborativos, comercio electrónico, portales web, etc.

La evolución que están sufriendo las aplicaciones web incide directamente en el incremento de la complejidad a la hora de diseñar, desarrollar, mantener y, por supuesto, garantizar la calidad de este tipo de aplicaciones Por tanto, la calidad en estos entornos es un concepto que tiene asociado una serie de atributos que se pueden observar directa o indirectamente, dando, la medida de éstos, un valor de estimación de la calidad total del sitio web.

Hoy en día todavía existen discusiones sobre cuáles son esos atributos, no existe un estándar ni un acuerdo general, pero en la literatura y propuestas recientes se plantean algunos conceptos en los que todos coinciden como son la usabilidad, funcionalidad, fiabilidad y eficiencia, entre otros. Cada uno de estos atributos se puede ver desde la perspectiva de los desarrolladores del sitio web y su funcionamiento y, por otro lado, desde el punto de vista del usuario y de cómo él percibe la funcionalidad global.

De todos estos atributos, en este capítulo se aborda la **Usabilidad**, uno de los factores de calidad más importantes ya que es imprescindible la calidad en uso para que un sistema web funcione adecuadamente y sea satisfactorio al usuario para el que va dirigido. De ahí su importancia como factor de calidad, especialmente en ambientes web.

8.2.2 Usabilidad: Métodos y Técnicas de Evaluación

Como se ha comentado en el apartado anterior, uno de los factores más importantes a la hora de caracterizar la calidad de una aplicación web es la usabilidad.

Este es el atributo más visible, puesto que es el referente que determina el grado de satisfacción del usuario con respecto a la aplicación Web y, por tanto, de ello depende que sea o no utilizada por el usuario.

La ingeniería del software suele centrarse en aquellos atributos del software que están relacionados con características internas del sistema, como son el rendimiento, la fiabilidad o la productividad, dejando de lado la perspectiva de los usuarios. En el caso del desarrollo de aplicaciones web, la usabilidad pasa a ser un factor muy importante debido a que estos productos están diseñados con el objetivo de que sean

usados por los usuarios y, por tanto, existe una gran interacción de los usuarios con el sistema Web.

El término usabilidad, según Bevan, Kirakowski y Maissel [2] aparece a comienzos de los años ochenta y es planteado desde tres puntos de vista:

- **Orientación al producto,** como una forma de medir términos de atributos ergonómicos del producto.
- **Orientación al usuario,** como una medida en términos del esfuerzo mental y de actitud del usuario frente al producto.
- **Rendimiento del usuario**, que establece relación con la medida de cómo el usuario interactúa con el producto, poniendo el énfasis en cómo de fácil es el producto de usar y cuál es su aceptabilidad, en el sentido de ser usado en el mundo real.

El estándar ISO 9126-1 [15] define la **usabilidad** como la capacidad de un producto software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

Esta definición hace énfasis en los atributos internos y externos del producto, los cuales contribuyen a su usabilidad, funcionalidad y eficiencia. Se aprecia claramente que la usabilidad depende no sólo del producto sino también del usuario. Por ello un producto no es en ningún caso intrínsecamente usable, sólo tendrá la capacidad de ser usado en un contexto particular y por usuarios particulares. La usabilidad no puede ser valorada estudiando un producto de manera aislada [3].

La usabilidad de un sistema no es un atributo que pueda especificarse independientemente del entorno de uso y de los usuarios concretos que vayan a utilizar el sistema, como se puede comprobar en la definición.

El estándar ISO 9241-11 [14] define la **usabilidad** como el grado en que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción.

Es una definición centrada en el concepto de calidad en el uso, es decir, se refiere a cómo el usuario realiza tareas específicas en escenarios específicos con efectividad, eficiencia y satisfacción.

Por **efectividad** se entiende la precisión y la plenitud con las que los usuarios alcanzan los objetivos especificados.

Por **eficiencia** se entiende los recursos empleados en relación con la precisión y plenitud con que los usuarios alcanzan los objetivos especificados.

Por **satisfacción** se entiende la ausencia de incomodidad y la actitud positiva en el uso del producto. Se trata de un factor subjetivo.

En la Figura 8.2, se puede ver la evolución del término usabilidad en los dos estándares ISO comentados. Como se puede observar, ambas definiciones de usabilidad son compatibles.

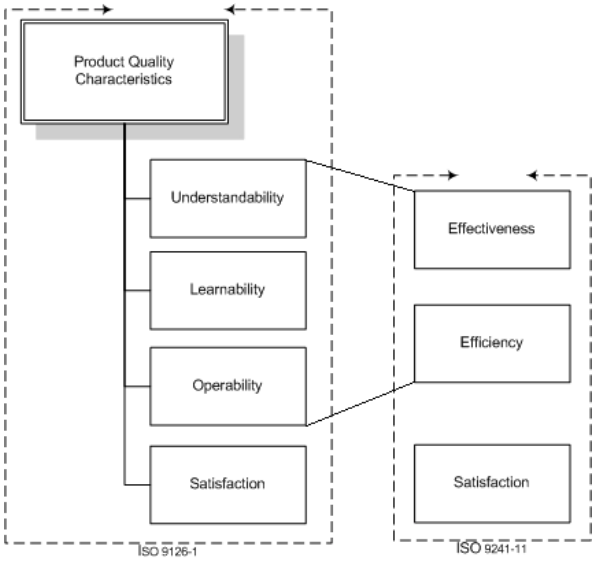


Figura 8.2: Usabilidad y estándares internacionales

A partir de las definiciones establecidas por la ISO, se pueden obtener una serie de principios fundamentales en los que está basada la usabilidad:

- **`Facilidad de aprendizaje:`** se refiere a la facilidad con la que nuevos usuarios pueden tener una interacción efectiva con el producto software, en este caso con la web.
- **`Flexibilidad`***:* se refiere a las distintas posibilidades con las que el usuario y el sistema pueden intercambiar información, teniendo en cuenta también la posibilidad de diálogo, la multiplicidad de opciones diferentes para realizar la tarea, la similitud con tareas realizadas anteriormente y la optimización entre el usuario y el sistema.
- **`Robustez:`** es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

Debido a esto la calidad se convierte en una calidad que depende de la percepción de los usuarios y puede surgir por tanto el concepto de **`calidad percibida`**. Este término se asocia al conjunto de características de un producto que aportan satisfacción a unos usuarios específicos. De este modo podría decirse que un producto sólo tendrá calidad en relación a sus propósitos establecidos.

El principal problema de la `usabilidad` reside en que es una cualidad demasiado abstracta para ser medida directamente. Por este motivo, J. Nielsen [25] la descompone en cinco atributos para poder estudiarla: `facilidad de aprendizaje, eficiencia, recuerdo en el tiempo, tasa de errores y satisfacción`. Además de esto, Nielsen agrega que la usabilidad tiene una estrecha relación con la aceptabilidad social y práctica del sistema, donde lo social se relaciona con la aceptación por parte de grupos de personas, mientras que lo

práctico incluye costos, soporte, confiabilidad y compatibilidad con sistemas existentes entre otras.

La evaluación de la usabilidad, tal y como se ha presentado, no es un tema nuevo, sino que data desde los comienzos del área *Human-Computer Interaction*, hace ya mas de cuarenta años. Sin embargo, los métodos para llevar a cabo evaluación de usabilidad de forma sistemática y estructurada son mucho mas recientes. Estos métodos son los llamados **Métodos de Evaluación de Usabilidad** (UEMs, *Usability Evaluation Methods*). Un UEM es `cualquier método o técnica empleada para llevar a cabo una evaluación de la usabilidad de cualquier sistema interactivo en cualquier etapa de su desarrollo.`

El objetivo de la evaluación es determinar si un sistema satisface un conjunto de requisitos ligados a la definición de usabilidad. El resultado de la evaluación puede ser una lista de problemas detectados o bien un conjunto de aspectos de diseño que mejoren la usabilidad del sistema.

En líneas generales, se pueden identificar dos grandes tipos de UEMs: empíricos y analíticos. Los **UEMs empíricos** conllevan la participación de los usuarios, y un equipo formado por evaluadores, observadores y expertos en tests, y normalmente utilizan un conjunto de tareas basadas en escenarios y técnicas como "Pensar en voz alta" (*Thinking Aloud)*. Normalmente se llevan a cabo en diferentes etapas del ciclo de vida del sistema, en prototipos o productos finales durante su operación y uso. Por otro lado, los **UEMs analíticos** no tienen acceso directo a los usuarios, pero incluyen un equipo de especialistas en usabilidad y desarrolladores de sistemas con menos conocimientos en temas de usabilidad, que normalmente utilizan un conjunto de directrices o heurísticas para estructurar y hacer más eficiente el proceso de inspección. Normalmente se llevan a cabo sobre prototipos o productos operativos pero también se pueden ejecutar sobre otros artefactos del proceso de desarrollo tales como modelos y se pueden llevar a cabo de forma manual o automática.

En los últimos años se han ido introduciendo diversos métodos para evaluar la usabilidad basados en diferentes paradigmas de evaluación formal, tales como test de usabilidad, estudios de campo, estudios predictivos, etc. Es más, en un esfuerzo de hacer las evaluaciones más fiables y menos subjetivas, se han desarrollado también sofisticados modelos matemáticos. Sin embargo, aunque la evaluación de la usabilidad ha sido tema de intensa investigación en la última década, todavía siguen existiendo numerosos e importantes problemas de usabilidad en todo tipo de productos software. Este hecho indica que la integración del desarrollo de software y la evaluación de la usabilidad en la industria de desarrollo de software no ha sido todo lo exitosa que debiera.

Como vemos existen bastantes métodos de evaluación de la usabilidad, pero estos, antes de ser excluyentes en su utilización, son en general complementarios. Como podemos apreciar en la tabla 8.1, cada uno de ellos puede ser útil en diferentes etapas del ciclo de vida de una aplicación software (R -> requisitos; D -> diseño; I->implementación; P -> pruebas; M -> mantenimiento).

Tabla 8.1. Métodos de evaluación de la usabilidad según las distintas etapas del ciclo de vida

Método de Evaluación	Fases ciclo de vida

8.3 Revisión y comparativa de diferentes modelos de calidad web

Como se ha visto hasta ahora, la calidad de un sistema Web se puede analizar de formas muy diversas en función de los atributos o características que se deseen garantizar o validar. Para intentar facilitar estos estudios y acotar el rango de atributos de calidad a tener en cuenta, se han ido definiendo algunos modelos de calidad que serán analizados en esta sección.

Los trabajos pioneros relacionados con factores de calidad, fueron realizados por McCall [McC, 77] y Boehm [Boe, 78]. Estos modelos tempranos son criticados por su carencia a la hora de decidir qué factores deberían incluirse en la definición de calidad y qué criterio debería asociarse con factores específicos, y además estos modelos no especifican explícitamente las métricas a usar en la capa inferior. Todos estos defectos hacen que estos modelos sean muy difíciles de aplicar. El modelo de Mc Call incluye once factores de calidad para describir tres aspectos de la calidad del producto: operación, revisión, transición. El modelo de Boehm usa siete factores de calidad para evaluar la capacidad de mantenimiento y la utilidad.

A la hora de definir modelos de calidad surgen distintas aproximaciones, la más extendida entre los desarrolladores es la aproximación **`Objetivo-Pregunta-Métrica`** (*Goal Question Metric*, GQM) [1] que se basa en el supuesto de que una organización que quiera medir de una forma decidida debe primero especificar las metas para si misma y sus proyectos, y después debe encontrar aquellos objetivos a

los datos que se desean para definir esas metas operacionalmente, y finalmente proveer un marco de trabajo para interpretar los datos respecto a las metas establecidas. La limitación principal de usar la aproximación GQM en un entorno de producción de software es la incapacidad de aislar los efectos de factores sencillos.

Otro de los modelos a destacar es el modelo ISO 9126, que, como ya se ha indicado, se basa en la definición de seis factores que caracterizan la calidad y en la asociación a cada uno de ellos de una serie de características y atributos que permiten describirlos de manera más precisa. Este modelo mejora el de Mc Call pero no tiene capas de criterios ni métricas [23]. Todos estos modelos suponen que los factores de calidad están en un nivel más alto y demasiado abstractos que no resultan fáciles de medir directamente.

A partir de estos modelos de calidad software surgieron los modelos de calidad web. A continuación se destacan algunos de los mas representativos, lo que permitirá tener una visión global de los modelos de calidad web existentes en la actualidad y sus características principales.

8.3.1 Modelos de calidad web

Un modelo de calidad es una representación de las características y atributos que debe tener una entidad o artefacto para cumplir de forma adecuada con las metas para las cuales se desarrolló. Como se ha visto, los modelos de calidad web surgen a partir de los modelos de calidad software por medio de aproximaciones como la GQM anteriormente comentada.

Bevan [4] establece tres tipos distintos de calidad en el desarrollo de software, que podrían ser extrapolados a las aplicaciones web. Estos tres tipos de calidad son la **`calidad interna`**, que es la medida interna de las propiedades estáticas del código por métodos de inspección, la **`calidad externa`**, que consiste en medir las propiedades del código en ejecución, y la **`calidad en uso`**, en la que hay que tener en cuenta a los usuarios que utilizan ese software y el ambiente de trabajo.

Como se ha apuntado anteriormente, los modelos de calidad existentes para aplicaciones software tradicionales, debido a sus carencias, deben ser adaptados al marco concreto de la web. Surge la necesidad, por tanto, de diseñar modelos de calidad que permitan especificar a los desarrolladores y evaluadores las características y atributos que deben tener los sistemas web para que cumplan con los requisitos deseados dado un contexto de uso y perfil de usuario concreto.

El desarrollo de sitios y aplicaciones web crece a pasos agigantados y sólo en estos últimos años es cuando se ha visto la necesidad de definir nuevos modelos que permitan evaluar la calidad en entornos web, y por esta razón no hay todavía ningún estándar en este campo concreto.

Aún así, se han realizado numerosos estudios y varias propuestas de modelos de calidad Web que serán analizadas a continuación.

`Web-site QEM`, o simplemente Web QEM, es la metodología propuesta en 1999 por Luis Olsina [27]. El objetivo de esta metodología es evaluar y comparar la calidad de aplicaciones web más o menos complejas. Web QEM surgió debido a la necesidad de una metodología integral, robusta y flexible para evaluar la calidad de sitios web.

Las características más destacadas de esta metodología son las siguientes:

- Está basada en modelos y métricas de calidad existentes, como el modelo ISO.
- Se basa en modelos de agregación de características y atributos, es decir, partiendo de unos atributos generales de calidad va añadiendo otros más específicos. Estos modelos pueden ser lineales o no lineales.
- La metodología se centra en el juicio de evaluadores expertos en el dominio.
- Favorece el seguimiento y la justificación de resultados.

La metodología de Olsina sigue la práctica común de describir la calidad software en términos de características como las definidas en el estándar ISO/IEC 9126-1 [15]. En esta propuesta se perciben atributos como propiedades que se pueden medir de una entidad y propone usar un modelo de calidad para especificarlos. WebQEM es un modelo de evaluación cuantitativa para medir la calidad de aplicaciones y sitios web.

Las características de calidad a medir que plantea Olsina son la funcionalidad, la confiabilidad, la eficiencia, la portabilidad, la capacidad de mantenimiento y la usabilidad. A su vez la usabilidad involucra la operabilidad, la comunicatividad, la estética y el estilo y la funcionalidad.

Web Tango (*Tool for Assessing Navigations and Organization)*, por otro lado, se trata de la metodología propuesta por Melody Ivory en su tesis doctoral [17], ampliada en estudios posteriores con el desarrollo de una herramienta automática.

Los objetivos principales de esta metodología son:

- Identificar un amplio conjunto de medidas de interfaz cuantitativas.
- Computar las medidas para un amplio conjunto de interfaces evaluadas.
- Derivar modelos estadísticos de las medidas y puntuaciones.
- Usar los modelos para predecir puntuaciones o una clasificación para nuevas interfaces.
- Validar las predicciones del modelo.

Para facilitar la discusión de los métodos de evaluación de usabilidad se propuso una taxonomía que consta de las siguientes cuatro dimensiones:

- **Clase de método:** prueba, inspección, consulta, modelado o simulación.
- **Tipo de método:** análisis de ficheros de registro, evaluación heurística, entrevistas, cuestionarios, análisis de tareas, etc.
- **Tipo de automatización:** manual, semi-automático, automático.
- **Nivel de esfuerzo***:* fácil de usar, efectivo y si se puede aplicar ampliamente.

El modelo de calidad de Ivory se centra en el desarrollo de una metodología que enmarque un modelo de evaluación totalmente automático para medir la calidad de sitios web. Esto permite a los diseñadores evaluar la usabilidad de sus diseños en las etapas tempranas del desarrollo con un bajo coste de recursos.

Ivory ha identificado y computado un número extensivo de medidas cuantitativas de sitios y páginas web y las ha usado para derivar modelos estadísticos de páginas y sitios altamente puntuados en una clasificación de calidad.

La metodología consiste en la computación de un conjunto de 157 medidas cuantitativas (métricas) a nivel de página y a nivel de sitio, como por ejemplo número de fuentes, imágenes y palabras, basadas en un estudio de recomendaciones de diseño de expertos reconocidos y diferentes tratados de usabilidad.

Otro de los modelos a destacar es el modelo de calidad para sitios web **WQM,** propuesto por Mario Piattini y Coral Calero [29]. Este es un modelo global de calidad

web que al igual que el resto de modelos revisados está motivado por la necesidad de desarrollar aplicaciones web con criterios de calidad para evitar un pobre rendimiento y la aparición de fallos.

El modelo surgió porque es necesario clasificar las métricas y los trabajos de investigación realizados sobre la web debido a que no existen ni estándares ni puesta en común de todas las iniciativas existentes hasta el momento. Es un modelo que no es excluyente de otros modelos de calidad existentes, sino que pretende ser aglutinador de los mismos. Define tres dimensiones:

- **Característica web.**
- **Característica de calidad.**
- **Proceso de ciclo de vida.**

En la Figura 8.3 se puede apreciar la representación gráfica de este modelo, con las 3 dimensiones que lo forman y las características de cada una de ellas.

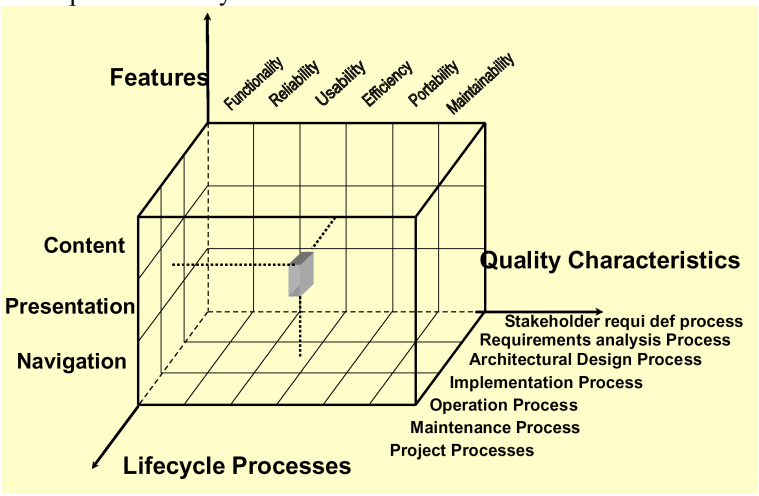


Figura 8.3. Modelo de calidad WQM

Como se ha podido apreciar en la revisión de los modelos y metodologías de calidad web, no existe ningún estándar en este ámbito. El único consenso que existe dentro de este ámbito son las características de calidad de interfaces software definidas por la organización de estándares internacional ISO 9126 [15], y las directrices de accesibilidad, de las que se hablará en el siguiente capítulo, sugeridas para asegurar la calidad web ampliamente aceptadas como son la *Web Accessibility Initiative* (WAI) [35] y la Sección 508 [30].

Aparte de esto, todos los modelos y metodologías revisadas siguen de alguna forma las fases definidas por el estándar ISO 9126-4 [16] para llevar a cabo la evaluación de calidad. Estas fases son: establecer el propósito de evaluación, especificar la evaluación, diseñar la evaluación y ejecutar la evaluación.

8.4 Métricas de usabilidad

Como se ha introducido en apartados anteriores, para poder medir la calidad de un sistema web de forma objetiva, es necesario definir una serie de métricas y medidas que permitan controlar la calidad de los diseños realizados. Además es necesario validar estas métricas de manera formal y empírica para que éstas sean realmente útiles. Hay multitud de estudios donde se definen métricas pero no hay un consenso en tales propuestas. En cuanto a validación, existe poco o ningún trabajo al respecto de las métricas propuestas, lo que hace poco fiable el uso de ciertas métricas y los estudios realizados por tanto son poco reutilizables.

Para entender de forma adecuada lo que son las métricas y en qué ámbito están definidas a continuación se dan algunas definiciones.

Por **medición** [8] se entiende el proceso por el cual se asignan números o símbolos a los atributos de las entidades en el mundo real, de tal manera que las definan de acuerdo con reglas claramente establecidas.

Es decir, gracias al proceso de medición, se obtiene una medida (número) a partir de la cual es relativamente fácil juzgar lo que se mide. Conociendo esto es fácil deducir que se deben medir atributos o características de la web, no la Web en sí. Por eso surge toda la discusión anterior acerca de cuáles son los atributos y subcaracterísticas que pueden caracterizar de mejor la calidad web. En consecuencia la definición de **medida** es la correspondencia de un dominio real a un mundo formal matemático. Es decir, pasar a datos numéricos lo que se observa en el mundo real.

Una medida es parte de una métrica, y no puede ser interpretada sin una unidad de medida y tipo de escala.

Una **métrica** es la correspondencia de un mundo real, o empírico, a un mundo formal, matemático. Por tanto, una métrica es un valor numérico o nominal asignado al atributo de un objeto o evento del mundo real, tangible o intangible, pudiendo ser un recurso (personas, software, hardware, oficina, etc.), proceso (requisito, diseño detallado, mantenimiento, etc.), producto (sitio o aplicación web, páginas, multimedia, programas, especificaciones, diseño, etc.) o producto en uso (aplicación web o componente web en uso).

Un **atributo** es la característica o propiedad de una entidad de tipo directo o indirecto, interno o externo. Por ejemplo, cantidad de enlaces rotos, tipo de color en los enlaces, etc.

Una métrica sólo tiene significado cuando se la compara con otros números. Según Steve Robinson, director de desarrollo web y tecnología en el grupo Econe de Xerox, una vez que se crean las métricas éstas empiezan a dar información acerca de preguntas que nunca se habían planteado antes de aplicarlas, y a partir de esas preguntas obtener respuestas que ayuden a mejorar la calidad de los sitios web [33].

En el estándar IEEE 1061 se puede leer que "El uso de métricas software reduce la subjetividad en la evaluación de calidad software al proveer una base cuantitativa para tomar decisiones acerca de la calidad del software. No

obstante, el uso de métricas software no elimina la necesidad del juicio humano en la evaluación del software".

8.4.1 Clasificación de las métricas de evaluación de calidad

Las métricas pueden clasificarse en base a dos consideraciones: según el grado de automatización; según el tipo de correspondencia.

Según el **grado de automatización** se puede encontrar la siguiente clasificación:

- **Métricas automatizables:** son aquellas que se pueden medir por medio de una herramienta automática que se usa para obtener los datos y realizar el cálculo.
- **Métricas semi-automatizadas:** son aquellas que se miden utilizando tanto técnicas manuales como automáticas.
- **Métricas manuales:** son aquellas que se realizan de forma totalmente manual.

Según el **tipo de correspondencia** las métricas se clasifican en:

- **Métricas directas:** son aquellas cuya correspondencia es directa, es decir, a partir de un atributo obtenemos un número, y este número se usa para responder preguntas.
- **Métricas indirectas:** son aquellas métricas en las que un atributo debe ser medido en combinación de otros atributos.

En el caso de sitios web un ejemplo de métrica directa es la cantidad de imágenes que tienen texto alternativo, ya que solamente hace falta medir la presencia de la etiqueta *ALT* en cada una de las imágenes de la web.

Un ejemplo de métrica indirecta es el porcentaje de enlaces rotos en un sitio, que se mediría dividiendo la suma de los enlaces rotos internos mas los enlaces rotos externos entre la cantidad total de enlaces, y multiplicado todo ello por 100 para obtener el porcentaje.

Además las métricas también pueden clasificarse como **métricas internas** y **métricas externas**.

Cuando se desarrolla un producto software, los productos intermedios se deberían evaluar usando métricas internas que miden propiedades intrínsecas que se pueden extraer inspeccionando el código. El propósito principal de estas métricas es asegurar que la calidad externa requerida se realiza. Las métricas internas proporcionan usuarios, evaluadores, probadores, y desarrolladores con el beneficio que ellos son capaces de evaluar la calidad del producto software y dirigir medidas de calidad pronto, antes de que el producto software sea ejecutable.

Las métricas externas usan medidas de un producto software derivadas de medidas del comportamiento del sistema del cual es parte, testeando, operando y observando el software ejecutable o el sistema. Antes de adquirir o usar un producto software debería ser evaluado usando métricas basadas en objetivos de negocio relativos al uso, explotación y gestión de el producto en un entorno organizacional y técnico específico. Las métricas externas proporcionan a los usuarios, evaluadores,

probadores y desarrolladores con el beneficio de que son capaces de evaluar el producto software durante las pruebas u operación.

En cualquier caso, como apunta Jim Sterne [33], no es necesario medir todo. Lo que se mida depende en parte de los objetivos del sitio Web. También apunta que la métrica más importante es aquella que ayuda a mejorar dicho sitio. No se puede por tanto generalizar en cuestión de métricas, ya que lo que mejora un sitio no tiene por qué mejorar todos. Es decir, hay que utilizar diferentes métricas para diferentes sitios, y lo que se mida depende de las metas que se quieran conseguir.

En el caso de aplicaciones web, las métricas que se han definido en diversos estudios han tenido que ver casi siempre con la calidad en uso, es decir, la percepción que tiene el usuario del producto en uso en un contexto determinado. De ahí que la mayoría de métricas propuestas toman como base características de alto nivel como la efectividad, la productividad y la satisfacción de usuario. Para poder medir estas características es necesario descomponerlas en atributos más sencillos y específicos que se puedan cuantificar por medio de números a partir de los cuales se puedan sacar conclusiones.

Olsina ha definido un modelo conceptual para la definición y explotación de métricas de calidad debido a que hasta ese momento los estudios que se habían realizado en este campo eran independientes y para dominios muy específicos, y por tanto a la hora de contemplarlos desde un punto de vista más general presentaban una serie de limitaciones. Este marco conceptual permite modelar las métricas de forma genérica independientemente de dominios software específicos.

Para otros autores, el estudio de las métricas conocidas de la web se realiza en base a sus atributos y medidas y éstas las clasifican en:

- **Propiedades del grafo web:** la Web puede representarse como una estructura de grafos donde las páginas web constan de nodos y enlaces. Este grupo de métricas cuantifica las propiedades estructurales de la web en una escala macro y microscópica.
- **Significación de la página web:** esta métrica formaliza las notaciones de calidad y relevancia de las páginas web con respecto a las necesidades de información del usuario. Las métricas de significación son utilizadas para tasar páginas en comparación, y que corresponden a búsquedas y tienen un impacto sobre la calidad de la búsqueda y recuperación en la web.
- **Características del uso:** se basan en patrones y acciones frecuentes en la forma en que los usuarios navegan por los recursos de la web, proporcionando invaluables pistas para mejorar el contenido, su organización y presentación del sitio. Esta métrica mide el comportamiento del usuario para estos propósitos.
- **Similitud de páginas web:** métrica que cuantifica la extensión de parentescos entre páginas web.
- **Búsqueda y recuperación en la página web:** éstas son métricas que capturan propiedades relacionadas con las necesidades de información, producción y consumo. Aquí los autores consideran las relaciones entre un número de regularidades observadas en la generación de información en la web.

En realidad en este caso no interesan este tipo de métricas, sino como se ha dicho aquellas que permitan evaluar la usabilidad de un sitio web. De ahí que las métricas con mayor importancia sean las métricas de calidad en uso y usabilidad que a

continuación se definen.

8.4.2 Métricas de Calidad en Uso

La calidad en uso es la visión que tiene el usuario de la calidad de un sistema software y es medida en términos del resultado de usar dicho software, y las propiedades del software por si mismo. La calidad en uso es el efecto combinado de las características de calidad software percibidas por el usuario. Posiblemente, y no a mucho tiempo, la calidad en uso de un producto sea el término llamado a sustituir a la, todavía extendida, usabilidad, como en su día ésta sustituyo al término **`interfaz amigable`**. La muestra de que esa mutación es factible ya ha sido recogida en el estándar ISO 9126-4, destinado al tratamiento de este concepto.

En ese mismo estándar, el ISO 9126-4, la `calidad en uso` queda definida en términos de eficiencia, productividad, seguridad y satisfacción, y se dice expresamente que la única diferencia entre ella y la usabilidad, en función de la definición que se da de ella en otros estándares (v. g. ISO 9241-11), es la característica de seguridad. Asimismo, junto con la definición de calidad de uso y de las características reseñadas aparecen métricas con las que estimar cada una de ellas, estas métricas se recogen en la siguiente tabla, junto con su método de evaluación y fórmula.

Tabla 8.1. Tabla de métricas propuestas en el estándar ISO 9126-4 (calidad de uso)

Nombre de la métrica	Método de evaluación	Fórmula
Métricas relacionadas con la efectividad		
Efectividad de la tarea	Test con usuarios	$M1 = \left\|1 - \sum A_i\right\|$ donde A_i es el valor proporcional de cada error en la salida obtenida al realizar la tarea
Completitud de la tarea	Test con usuarios	$X = A/B$ donde A es el número de tareas completadas y B el número de tareas intentadas
Frecuencia de error	Test con usuarios	$X = A/T$ donde A es el número de errores cometidos por el usuario y T es el tiempo o número de tareas
Métricas relacionadas con la productividad		
Tiempo en llevar a cabo la tarea	Test con usuarios	$X = T_a$ donde T_a es el tiempo invertido en la tarea
Eficiencia de la tarea	Test con usuarios	$X = M1/T$ donde M1 es la efectividad de la tarea y T es el tiempo invertido en llevarla a cabo
Productividad económica	Test con usuarios	$X = M1/C$ donde M1 es la efectividad de la tarea y C es el coste total de la tarea
Proporción de tiempo productivo	Test con usuarios	$X = T_a/T_b$ donde T_a es el tiempo productivo y T_b es el tiempo en completar la tarea, incluyendo tiempo en que se consulta la ayuda, se subsanan errores o se determina cómo realizarla.
Eficiencia relativa del usuario	Test con usuarios	$X = A/B$ donde A y B son la eficiencia en llevar a cabo la tarea por usuarios no expertos y expertos respectivamente.

Métricas relacionadas con la seguridad		
Seguridad y salud del usuario	Registro de utilización	$X = 1 - A/B$ donde A es el número de problemas de salud reseñados por los usuarios y B es el número total de usuarios
Seguridad de la gente afectada por el uso del sistema	Registro de utilización	$X = 1 - A/B$ donde A es el número de personas escogidas al azar y B es el número total de personas potencialmente afectadas por el sistema
Daños económicos	Registro de utilización	$X = 1 - A/B$ donde A es el número de ocurrencias de daños económicos y B es el número total de usos
Daños software	Registro de utilización	$X = 1 - A/B$ donde A es el número de ocurrencias de defectos software y B es el número total de utilizaciones
Métricas relacionadas con la satisfacción		
Escala de satisfacción	Cuestionarios al usuario	$X = A/B$ donde A es el valor de escala producido por el cuestionario y B es la media de la población
Cuestionario de satisfacción	Cuestionarios al usuario	$X = \sum A_i / n$ donde A_i es la respuesta a una cuestión y n es el número de respuestas
Aceptación del producto	Observación	$X = A/B$ donde A es el número de veces que funciones de software específico son utilizadas y B es el número de veces que las mismas son invocadas

Junto a la información recogida en la tabla, en dicho estándar también se dan directrices generales con las que poder llevar a cabo un experimento de evaluación de la calidad en uso de un producto software, así como con la realización del informe e interpretación de los datos recabados con su realización.

Esta amplia visión de calidad tiene tradicionalmente el factor de ergonomía, que se refiere a los factores en el entorno social y físico que influye en la extensión en la cual el usuario puede alcanzar sus metas. Todos estos elementos del sistema de trabajo determinan cómo se comporta la gente y si tienen éxito en sus tareas. La salida del sistema de trabajo se puede medir como efectividad, productividad, y satisfacción de los usuarios.

Según Constantine y Lockwood [6] las métricas prácticas deberían:

- Ser fáciles de calcular e interpretar.
- Aplicarse a prototipos en papel y modelos de diseño.
- Tener una base racional fuerte y de conceptos simple.
- Tener suficiente sensibilidad y capacidad para discriminar entre diseños.
- Ofrecer guía directa para el diseño.
- Predecir efectivamente la usabilidad real en la práctica.
- Indicar directamente calidad relativa de diseños.

Una base conceptual sencilla significa que los diseñadores pueden entender el fundamento de una métrica y ver lo que hace un diseño mejor que otro en términos de la misma. Una buena base conceptual facilita a los diseñadores entender cómo se reflejan las diferencias entre los diseños según las diferencias en los niveles de medida. Idealmente la medida, y el proceso de medida en sí, deberían guiar al diseñador hacia diseños mejores. Las métricas que se pueden usar en prototipos o modelos de diseño no funcionales serán más útiles que aquellas que requieren sistemas completos o parcialmente en funcionamiento.

8.5 Integración de la calidad en el proceso de desarrollo

Debido a la importancia que supone el factor usabilidad en la calidad del software, distintos autores han resaltado la importancia de integrarla dentro del ciclo de vida del desarrollo de sistemas, desde las fases más tempranas de análisis de requisitos hasta las fases finales de pruebas y evaluación una vez lanzado el producto, como ya se indicó en el capítulo 1. Durante el ciclo de vida, cuanto más tarde se detecte un problema, más caro resulta resolverlo. Por tanto, cuanto antes se integre la usabilidad en el proceso de desarrollo, mejores serán los resultados. Sin embargo, existen grandes dificultades a la hora de integrar la ingeniería de usabilidad dentro de los procesos tradicionales de ingeniería del software, de hecho, este es un tema que todavía no se ha resuelto y es objeto de investigación actualmente.

Los ingenieros del software tradicionalmente han entendido la usabilidad de forma análoga a otros atributos que se prueban en test de garantía de calidad, normalmente al final del proceso de desarrollo. Esta situación ha llevado a la aplicación de técnicas de usabilidad muy tarde en el ciclo de desarrollo cuando los principales problemas de usabilidad son demasiado costosos de arreglar. Según autores como Trenner [34], los principales problemas de usabilidad son fácilmente detectables en las primeras fases de desarrollo, y por tanto hay una tendencia a integrar la ingeniería de usabilidad de una forma más compacta.

Tal integración no es sencilla, debido a que hasta ahora ambos procesos de desarrollo discurrían casi independientemente, llevados a cabo por equipos distintos.

Para complicar más la cuestión, hay que tener en cuenta que la usabilidad es uno más de los muchos atributos de calidad que los ingenieros de software deben considerar a la hora de desarrollar un sistema, por ejemplo, la funcionalidad, rendimiento, fiabilidad, seguridad, mantenibilidad, etc. Además, en condiciones reales, dadas las restricciones de costes y tiempo, los desarrolladores deben priorizar aquellos atributos de calidad que consideran más importantes para un sistema concreto. Con lo cual, es extremadamente difícil, sino imposible, integrar completamente todos los aspectos de usabilidad presentados anteriormente, junto con el resto de atributos de calidad en un mismo sistema. Para dar solución a este tema, han surgido algunas propuestas que analizamos a continuación, además de la de Mayhew [18] presentada en detalle en el capítulo 1.

Constantine y Lockwood [6] proponen un **Diseño Centrado en el Uso** que puede considerarse como un método que define las técnicas a usar, más que un proceso de desarrollo tal y como se entiende en ingeniería del software. Este diseño centrado en el uso consiste en un conjunto de actividades coordinadas orientadas a la usabilidad. Entre ellas se incluye el modelado de tareas, modelado de interfaz, etc. La característica más importante que introducen los autores es el concepto de **Casos de uso esenciales**. Ésta es una variación de la definición original de casos de usos y es la base para este diseño centrado en el uso.

Costabile [7] propone hacer frente al problema de la usabilidad integrando métodos centrados en el usuario dentro del proceso software. Se basa en tres conceptos:

- Análisis de usuarios y tareas.
- Diseño e implementación iterativo utilizando prototipos.
- Evaluación de dichos prototipos por parte de usuarios.

La principal idea de su propuesta comienza con el ciclo de vida en cascada. Propone una modificación de éste incluyendo dos nuevas actividades orientadas a la usabilidad. La primera está relacionada con análisis de tareas y usuarios, escenarios y especificaciones de interfaz de usuario; la segunda está relacionada con prototipado y pruebas. Además, define la posibilidad de volver a una etapa anterior desde cualquier otra etapa del ciclo de vida. Estos retrocesos en la cascada junto con las actividades de prototipado y pruebas ocurren dos veces en el ciclo de vida.

Una última propuesta a destacar es la de Stephanidis [32], que propone la *Unified User Interface Development Methodology*. Esta propuesta se caracteriza por su aproximación sistemática para hacer frente a la diversidad de los grupos de usuarios posibles, las tareas y los entornos de uso, mediante la adaptación automática de la interfaz de usuario. El objetivo es identificar y enumerar posibles diseños alternativos apropiados para diferentes usuarios y contextos de uso; identificar abstracciones y fusionar alternativas en patrones de diseño abstractos y finalmente racionalizar el espacio de diseño.

Pero como se puede apreciar de todas las propuestas comentadas, desafortunadamente no existe una definición estándar de cómo la usabilidad debe ser incluida dentro del ciclo de vida del software. Quedan todavía muchas lagunas sin cubrir en cuanto a la integración de la usabilidad dentro de los procesos de desarrollo de software. Algunas de estas carencias se pueden resumir en una falta de iteratividad real, falta de lenguajes de modelado de interfaces de usuario, falta de formalización y definición de los diferentes artefactos a construir, y sobre todo, falta de aproximaciones realmente centradas en el usuario.

8.5.1 IDEAS: Un marco metodológico orientado a la usabilidad

B. Shneiderman [31] explica que, dentro de la "ingeniería de la usabilidad", una de las bases para alcanzar un alto nivel de calidad es utilizar métodos de desarrollo iterativos. Estos métodos permiten hacer uso de prototipos, consiguiendo así obtener información de las reacciones de los usuarios.

J. Nielsen [25] establece que la mejor forma de aplicar con éxito criterios de usabilidad es poniendo el máximo esfuerzo antes de que comience el proceso de diseño de la interfaz. En este sentido, lo más importante es llevar a cabo un estudio previo que permita a los desarrolladores conocer a los usuarios. Este conocimiento se basa, no sólo en el estudio de las características o habilidades individuales de los usuarios, sino también en las tareas que deben ejecutar con el sistema. En este sentido es muy importante que los usuarios puedan llevar a cabo evaluaciones (v.g. tests de usabilidad) sobre un prototipo. De este modo, se pueden ir obteniendo mejoras en sucesivas versiones del sistema.

Por estas y otras razones, IDEAS [20] incluye modelos de alto nivel que permiten `conocer a los usuarios` y conocer las tareas que éstos necesitan ejecutar. Para ello, incluye modelo de casos de uso, modelo de tareas y modelo de usuario. Los dos primeros permiten describir de forma precisa la funcionalidad del sistema y las tareas de usuario. El último describe las características particulares de los tipos de usuarios diferentes que pueden interactuar con el sistema.

Después de conocer tanto a los usuarios como las tareas que ejecutan, se establecen

los objetivos de usabilidad que el sistema debe alcanzar. La definición precisa de estos criterios permitirá posteriormente llevar a cabo las diferentes pruebas de usabilidad asociados a tales criterios. En esta etapa, se define un modelo de calidad [24] que recoge y pondera los criterios de usabilidad establecidos y las métricas más adecuadas para medirlos.

En las últimas fases del desarrollo, a la hora de construir el modelo de presentación, el diseñador teniendo en cuenta el modelo de calidad inicialmente definido, selecciona las heurísticas y directrices de usabilidad correspondientes a los criterios definidos en el modelo de calidad. A partir de los modelos de presentación, se generan prototipos de la interfaz de usuario. Un conjunto de usuarios finales evalúan estos prototipos utilizando diferentes métodos y test de evaluación de usabilidad con el objeto de satisfacer exclusivamente los criterios definidos en el modelo de calidad establecido inicialmente.

Finalmente, todas las fases y modelos desarrollados están incluidos en un proceso de desarrollo iterativo. Esta forma de hacer permite llevar a cabo un proceso de feedback que nos permite introducir la información recogida en las pruebas realizadas por los usuarios en iteraciones previas a lo largo del proceso. De este modo, la usabilidad de los desarrollos mejora notablemente.

En la Figura 8.4 se ilustra este proceso iterativo de desarrollo orientado a la usabilidad.

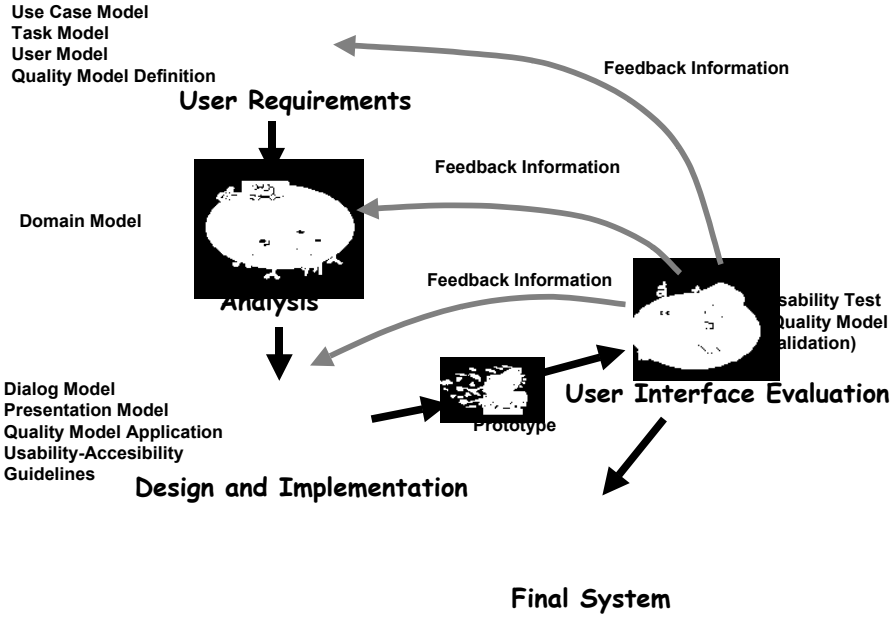


Figura 8.4: IDEAS: proceso de desarrollo iterativo orientado a la usabilidad.

Además, es importante resaltar que este proceso de desarrollo es independiente de la plataforma final donde se ejecute la aplicación. Bien puede ser un entorno web, una aplicación de escritorio en un ordenador personal, un PDA, *Smart Phone* o cualquier otra plataforma, ya que este tipo de decisión se hace sólo en la última fase de

desarrollo de esta metodología; es decir, la idea es que el mismo diseño pueda ser utilizado para diferentes implementaciones en diferentes dispositivos.

A modo de ejemplo, y sin entrar en detalles, ya que no es el objetivo de este capítulo, a continuación se muestra un pequeño fragmento de un caso de estudio desarrollado con la herramienta **IDEAS-CASE** [21] que soporta el proceso descrito anteriormente. El ejemplo está basado en una aplicación web para un Sistema de Revisión de Conferencias. Las siguientes figuras muestran parte del proceso de desarrollo para una de las funcionalidades del sistema, la identificación del usuario para acceder al sistema, empezando por el modelo de casos de uso, hasta la generación automática de la aplicación final.

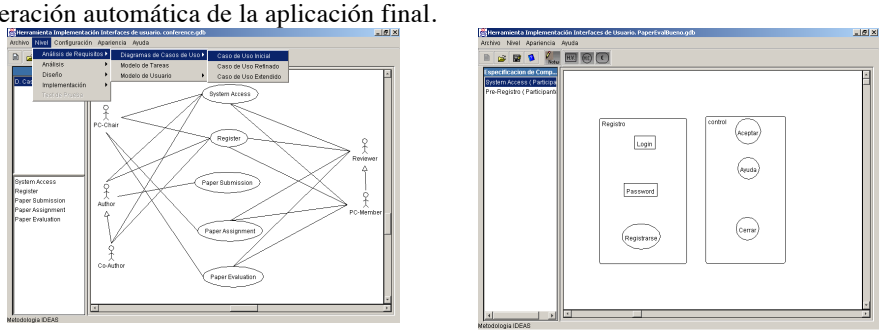


Figura 8.5. Modelo de Casos de Uso y Diagrama de Especificación de Componentes.

A partir de los Objetos de Interacción Abstractos definidos en el Diagrama de Especificación de Componentes (Figura 8.5) se construye el Modelo de Presentación, en el que se seleccionan los Objetos de Interacción Concretos mas adecuados como muestra la imagen de la izquierda de la Figura 7. A partir de este modelo se genera automáticamente la interfaz final, tal y como se muestra en la imagen de la derecha de la Figura 8.6.

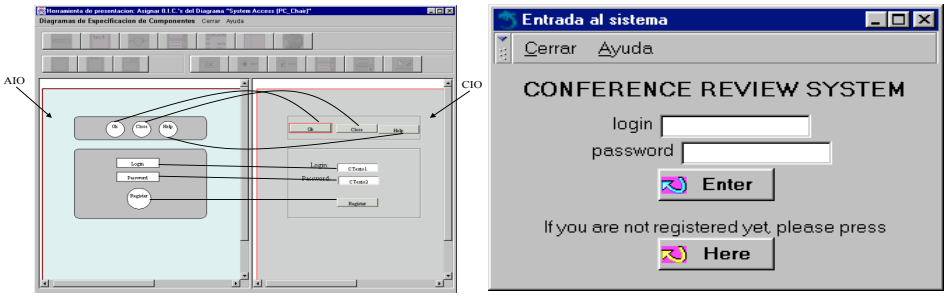


Figura 8.6. Modelo de Presentación e interfaz gráfica final generada

La evaluación de la calidad en esta metodología se integra en el proceso de desarrollo propuesto e incluye las cuatro fases propuestas por la ISO 14598 [14]: establecimiento de requisitos de evaluación; especificación de la evaluación; diseño de la evaluación; y ejecución de la evaluación. Por otra parte, el proceso de evaluación se realiza de manera incremental, es decir, se realiza la evaluación del software conforme se va desarrollado éste. Siguiendo la propuesta de la ISO 14598, el proceso parte de un modelo de calidad previamente definido en la fase de requisitos

que permite describir los factores más importantes a tener en cuenta en el desarrollo de una determinada aplicación. Este modelo de calidad se define en base al conocimiento del contexto de uso (entorno, usuarios, plataforma) y es el que permite decidir las métricas y los métodos de evaluación más convenientes para establecer las medidas oportunas sobre el sistema. Por otra parte, para asegurar una mayor calidad, descrita esta en base a la usabilidad, dentro del proceso de diseño se tienen en cuenta una serie de guías de estilo de diseño que mejoran la usabilidad y accesibilidad de las aplicaciones. Fruto del desarrollo se producen una serie de artefactos software que son analizados según las técnicas de evaluación y las métricas seleccionadas previamente. En función de los resultados obtenidos en estos procesos de evaluación se pueden proponer modificaciones en el producto software tendentes a mejorar su usabilidad. Como puede apreciarse en la figura 8.4, dichas modificaciones pueden ligarse a las distintas fases del proceso de desarrollo propuesto.

8.6 Conclusiones

En este capítulo se ha abordado el tema de la calidad en el desarrollo de aplicaciones web. Como hemos visto, la calidad es un concepto muy amplio que abarca numerosos y diferentes factores. Por ello, se ha centrado en uno de los factores más importantes a la hora de garantizar la calidad en entornos web, qué es la Usabilidad.

Los sistemas web construidos con criterios de usabilidad tienen numerosos beneficios, tales como mejora de la productividad, reducción del tiempo de aprendizaje y coste y un incremento notable en la autonomía de los usuarios finales.

De hecho, numerosos estudios [19] demuestran que el 80% de los costes de mantenimiento -lo que a su vez representa el 80% de los costes totales del desarrollo- se deben a problemas de interacción usuario-sistema y no a problemas técnicos. Además, los estudios indican que el 64% de estos costes están relacionados con problemas de usabilidad.

Esta situación resalta la importancia de la usabilidad como factor crítico de la calidad de los sistemas y justifica la necesidad de incorporarla antes, durante y después del proceso de de desarrollo de software. Pero diseñar sistemas web usables no es tarea fácil. Para afrontar este problema, tenemos que tener en cuenta que la aplicación será utilizada por usuarios con diferentes habilidades, objetivos y preferencias. Este hecho resulta especialmente importante cuando hablamos de aplicaciones basadas en web, donde la heterogeneidad de los usuarios es mucho mayor. En cualquier caso, el desarrollo de cualquier sistema software y especialmente el desarrollo web debe estar guiado por un proceso centrado en el usuario y orientado a la usabilidad.

Por todo ello, en este capítulo se ha hecho una completa revisión al tema de la calidad en entornos web, centrándose en el atributo de usabilidad principalmente. Se han presentado los métodos más extendidos de evaluación de usabilidad y la importancia que tienen los modelos de calidad para facilitar la definición de los atributos a tener en cuenta. En este sentido se ha hecho una revisión de los modelos de calidad más destacados hasta el momento, así como el uso de métricas como forma de

medir objetivamente el grado de usabilidad alcanzado.

Finalmente se ha visto que es necesario incorporar criterios de usabilidad dentro del propio proceso de desarrollo software desde las etapas más tempranas. Actualmente existen pocas propuestas que aborden este tema y es necesario avanzar en esta línea. Por ello, aparte de hacer una revisión de algunas propuestas planteadas en la literatura, se ha presentado un marco metodológico que incorpora un modelo de calidad al principio del proceso de desarrollo como mecanismo que permite definir y validar los criterios de usabilidad requeridos. Este proceso de desarrollo no sólo permite generar aplicaciones con alto nivel de usabilidad, sino que su naturaleza iterativa y la definición de diferentes niveles de abstracción permiten generar diferentes implementaciones en diferentes plataformas a partir de las mismas especificaciones de diseño, en todos los casos atendiendo a las características de usabilidad correspondientes según el caso.

Agradecimientos

Este trabajo ha estado parcialmente soportado por los proyectos TIN2004-08000-C03-01, financiado por el Ministerio de Educación y Ciencia, y PBC-03-003, financiado por la Consejería de Educación de la Junta de Comunidades de Castilla-La Mancha.

Referencias

1. Basili, V.R., Caldiera, C., Rombach, H.D., (1994). Goal Question Metric Paradigm, Encyclopedia of Software Engineering, Vol. 1, John Wiley & Sons, pp. 528532.
2. Bevan, N., Kirakowski, J., Maissel, J. (1991). What is Usability. Proceedings of the 4th International Conference on HCI, Stuttgart, September 1991.
3. Bevan N.; Macleod, M., (1994). *Usability measurement in context*. Behaviour and Information Technology, 13, 132-145.
4. Bevan N and Azuma M, (1997). *Quality in use: Incorporating human factors into the software engineering lifecycle*, in Proceedings of the Third IEEE International Software Engineering Standards Symposium and Forum (ISESS'97), p169-179.
5. Boehm, B.; Brown, J.R.; Kaspar, J.R. (1978). *Characteristics of Software Quality*, TRW Series of Software Technology.
6. Constantine, L. and Lockwood, L. (1999). *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, New York.
7. Costabile, M.F. (2001). *Usability in the Software Life Cycle*. Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing, pp. 179-192. Singapore.
8. Fenton, N.E. and Pfleeger, S.L. (1997). *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press.
9. Gilb, T., (1969). Weighted Ranking by Levels, (IAG Journal, Vol 2 (2), 1969, pp. 7-22.
10. Ginige, A., Murugesan, S. (2001). Web Engineering: An Introduction. IEEE Multimedia. Enero-Marzo 2001.
11. ISO/IEC 9126 International Standard (1991). Information technology – Software product evaluation – Quality characteristics and guidelines for their use.
12. ISO 8402, (1994). Quality management and quality assurance – Vocabulary.

13.ISO/IEC 14598-1, (1998). Information Technology - Evaluation of Software Products - Part 1 General guide.
14.ISO 9241-11, (1998). Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11 Guidance on usability.
15.ISO/IEC FDIS 9126-1 (2000). Software Engineering - Product quality - Part 1: Quality model.
16.ISO/IEC 9126-4. (2001). Software Engineering – Software Product Quality- Part 4: Quality in use metrics, 2001.
17.Ivory, M.Y. (2001). An Empirical Foundation for Automated Web Interface Evaluation. Ph. D. Thesis.
18.Mayhew, D.J. (1999). The Usability Engineering Lifecycle. Morgan Kaufmann, San Francisco, CA.
19.Landauer, T.K. The Trouble with Computers: Usefulness, Usability and Productivity. MIT Press, 1995.
20.Lozano, M. (2001). Entorno Metodológico Orientado a Objetos para la Especificación y Desarrollo de Interfaces de Usuario. Tesis Doctoral. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia.
21.Lozano, M., Molina, J.P., Montero, F., González, P., Ramos. I. A Graphical User Interface Development Tool. Proceedings of the 6th Anual Conference on Human Computer Interaction (HCI'02) Ref.: ISBN: 1-902505-48-4, Londres, Inglaterra, 2002.
22.Mc Call, J.A., Richards, P. G., and Walters, G. F. (1977). Factors in Software Quality, Vols. I, II and III (NTIS AD/A-049 014/015/055), Springfield: NTIS 1977.
23.Mei, H., Xie, T., Yang, F., (2002). A model-based approach to object-oriented software metrics. Journal of Computer Science and Technology. Volume 17, Issue 6 (November 2002). Pages 757- 769. 2002.
24.Montero, F., Lopez-Jaquero, V., Lozano, M.; González, P. (2003) A Quality Model For Testing the Usability of Web Sites. International Conference on Human-Computer Interaction HCII´03. Creta. Grecia.
25.Nielsen, J. (1993). Usability Engineering. Morgan Kaufmann.
26.Nielsen, J., (2000). Designing Web Usability: The Practice of Simplicity. New Riders Publishing.
27.Olsina, L. (1999) Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web. PhD. Thesis, Universidad Nacional de La Plata, Argentina.
28. Olsina, L., Rossi, G. (2001). A Quantitative Method for Quality Evaluation of Web Sites and Applications. IEEE Multimedia Magazine.
29.Piattini, M. Calero C., Ruiz, J.(2003). A Three Dimensional Web Quality Mode.l J.M.Cueva Lovelle et al. (Eds.): ICWE 2003, LNCS 2722, pp. 384-385.
30.Section 508. http://www.section508.gov/
31.B. Shneiderman. (1998). Designing the User Interface. Strategies for Effective Human-Computer Interaction. Addison Wesley,.
32.Stephanidis, C., Savidis, A. (2001). Universal Access in the Information Society: Methods, Tools and Interaction Technologies. UAIS 1: 40-55.
33.Sterne, J. (2002). Web Metrics Proven Methods for Measuring Web Site Succes. Wiley Publishing Inc.
34.Trenner L., Bawa J. (1998). The Politics of Usability. Springer-Verlag.
35.Web Accessibility Initiative of W3C (1999). http://www.w3c.org/WAI

9 Ingeniería de la accesibilidad a la web

Julio Abascal, Myriam Arrue, Nestor Garay, Juan Miguel López, Markel Vigo

Laboratorio de Interacción Persona-Computador para Necesidades Especiales
Universidad del País Vasco-Euskal Herriko Unibertsitatea
Manuel Lardizabal 1, E-20018 Donostia
[julio.abascal, myriam, nestor, juanmi, markel] @si.ehu.es

9.1 Accesibilidad a la web

Cada vez se provee más información y servicios a través de sistemas basados en la informática y la telemática. Muchos de ellos son realmente útiles y necesarios para facilitar la vida diaria de las personas. Sin embargo, muchas personas con restricciones físicas, sensoriales o cognitivas suelen encontrar serios problemas para acceder a ellos. Aunque las personas con discapacidad son un claro ejemplo de la exclusión que comporta la falta de accesibilidad, no son las únicas que la experimentan. Similares restricciones encuentran quienes usan equipos con visualizadores de pequeño tamaño o definición, conexiones de bajo ancho de banda, navegadores especiales, etc.

Volviendo al caso concreto de las personas con discapacidad, es necesario subrayar que, dadas sus limitaciones, los servicios ofrecidos a través de la web pueden ayudar a evitar algunas de sus restricciones, por lo que resultan incluso más necesarios para ellas que para el resto de la población. Algunos países han reconocido esta necesidad como un derecho civil y lo han protegido y regulado a través de la ley.

A pesar del reconocimiento público de la importancia de la accesibilidad y de la protección legal, muchos diseñadores encuentran más dificultoso el diseño de páginas web accesibles, seguramente debido a la escasez de metodologías y de herramientas adecuadas para esta tarea. En este capítulo se tratará de demostrar que el esfuerzo extra de desarrollar sitios web accesibles viene compensado por una mayor usabilidad para todos los usuarios [1] y por una mayor visibilidad para los robots de búsqueda. Tal como escribe S. Pemberton:

"¿Cómo puedo justificar el coste extra para un porcentaje tan pequeño de público? La respuesta es Google. […] Y la visión de Google es exactamente la misma que la de una persona ciega, Google es un usuario ciego —un usuario ciego millonario, con millones de amigos que escuchan cada una de sus palabras. Si un usuario ciego no puede ver tu sitio, tampoco Google puede…" [2]

9.1.1 Introducción a la accesibilidad

El objetivo de la **`accesibilidad a la web`** consiste en garantizar que las aplicaciones web puedan ser accedidas y usadas por todos los usuarios potenciales, independientemente de las limitaciones propias del individuo o de las derivadas del

contexto de uso. Por tanto, incluye el uso de cualquier tipo de navegador (actual, antiguo, de propósito especial), de cualquier tipo de ordenador (de baja o alta capacidad de procesado, baja o alta definición de pantalla, cualquier tamaño de *display*, etc.), o de cualquier elemento de acceso alternativo (televisión, dispositivo móvil, etc.), de cualquier tipo de conexión (con bajo o alto ancho de banda), y por personas con todo tipo de características físicas, sensoriales o cognitivas.

9.1.1.1 La diversidad humana. Las personas presentan una gran variedad de características físicas y cognitivas, algunas de las cuales pueden estar restringidas, de manera permanente o transitoria, en comparación con "la mayoría de la población", desde el nacimiento o a causa de una enfermedad o accidente. Para cada una de las funcionalidades humanas se define un rango convencionalmente considerado normal y se considera "especiales" a las personas que se sitúan fuera de ese rango. Concretamente a las personas que presentan restricciones a partir de cierto nivel se les considera discapacitadas. Hoy en día cada vez es más manifiesta la arbitrariedad de la definición del límite entre la normalidad y la discapacidad, que varía de un país a otro dependiendo de la cultura, de la tecnología e incluso de la riqueza disponible. De tal modo que una persona puede ser considerada como discapacitada, o no serlo, dependiendo del país en el que vive [3]. Tal arbitrariedad tiene que hacer comprender lo inapropiado del diseño de tecnología válida para unos, pero excluyendo a otros a los que se considera discapacitados.

Algunas de las discapacidades que afectan en mayor o menor medida a la accesibilidad a la web son:

- discapacidades visuales: ceguera, baja visión, daltonismo, acromatopsia, vista, cansada, etc.,
- discapacidades auditivas: sordera, pérdida auditiva, etc.,
- discapacidades físicas: principalmente problemas motores en los miembros superiores,
- discapacidades del habla: pérdida del habla, baja inteligibilidad de la misma, etc.,
- discapacidades cognitivas: dislexia, discalculia, déficit de atención, problemas de memoria, etc.
- limitaciones físicas, sensoriales o cognitivas ocasionas por el envejecimiento.

Estas limitaciones afectan a un porcentaje elevado de la población y van aumentando con la edad, por los cambios que se producen en la visión, oído, memoria, funciones motoras, etc. Los problemas de accesibilidad de las personas con discapacidad y de los ancianos, deben tratarse hoy en día como una faceta más de la diversidad humana. De hecho, tienen gran relación con las dificultades que experimentan los usuarios que tratan de acceder a la web en condiciones especiales. Por ejemplo, las personas que acceden con las manos ocupadas, mientras están realizando otra tarea como conducir o manipular equipamiento, o acceden desde lugares especiales, poco iluminados o ruidosos, o utilizan equipamiento especial, tal como dispositivos móviles con visualizadores pequeños. De este modo la población que se vería favorecida por el diseño accesible no es tan pequeña como se podría pensar en un principio.

Ante la gran variedad de discapacidades, los diseñadores de sitios web a los que se pide un diseño accesible suelen reaccionan negativamente. Argumentan que la

accesibilidad limita la libertad del diseño tanto desde el punto de vista técnico como estético. Además se sienten agobiados por la necesidad de conocer y aplicar las restricciones de una variedad enorme de usuarios. En los siguientes apartados se tratará de probar que el uso de pautas de diseño accesible no limita la libertad ni las posibilidades del diseñador y que, además, hace innecesario conocer en detalle las características de cada tipo de usuario.

9.1.1.2 Tecnología Asistencial. Las personas con restricciones físicas o sensoriales suelen encontrar dificultades para utilizar los dispositivos de interacción más comunes (teclado/ratón y pantalla). La **`Tecnología Asistencial`** se dedica al desarrollo de todo tipo de dispositivos y *software* destinado a facilitar el uso del ordenador a estas personas. Para ello se diseñan dispositivos y procedimientos de interacción alternativos que aprovechas las capacidades de cada persona, lo que se conoce en algunos entornos como desarrollo de *software* y *hardware* adaptativo o alternativo [3].

La tecnología asistencial, denominada en inglés *Assistive Technology*, se orientó inicialmente al desarrollo de soluciones particulares para las necesidades concretas de cada usuario a través de un proceso frecuentemente artesanal. Este enfoque resultaba caro y lento, y sus resultados eran difícilmente generalizables a otros usuarios con parecidas necesidades. Además lo rápidos cambios tecnológicos, tanto del *hardware* como del sistema operativo y del *software* de usuario, hacían que muchas de las adaptaciones resultaran obsoletas al poco tiempo y requirieran comenzar el trabajo desde el principio para la nueva tecnología disponible en el mercado [4]. El deseo de los usuarios con discapacidad de utilizar *software* comercial más que programas especiales impulsó la aparición de un nuevo enfoque de diseño basado en el aprovechamiento de las funcionalidades presentes en cada usuario, más que en la evitación de las restricciones. Actualmente la Tecnología Asistencial se centra en el diseño de dispositivos que garanticen a cada usuario la posibilidad de usar programas y equipamiento estándar, desarrollado dentro de la filosofía del "diseño para todos" [5]. Los productos de la Tecnología Asistencial incluyen teclados alternativos (entrada por barrido, teclados reducidos, emuladores de ratón, etc.), reconocimiento del habla, predictores de palabras, pantallas táctiles, líneas *braille*, ampliadores y lectores de pantalla, navegadores de texto y o voz, etc. [6, 7, 8].

9.1.1.3 Diseño para todos. Los productos de la Tecnología Asistencial (dispositivos de interacción específicos para las personas con determinadas discapacidades) resultan inútiles si el equipamiento comercial no sigue los principios del diseño universal. Tal como su nombre indica, el **`Diseño para Todos`** es una filosofía de trabajo que trata de que los productos sean utilizables por el máximo posible de usuarios. En el caso de la tecnología telemática pretende que los servicios y sistemas sean accesibles independientemente de los dispositivos de interacción que se usen. Haciendo un símil con la Arquitectura, el diseño para todos garantiza (v.g. mediante rampas) la accesibilidad a los usuarios provistos de la tecnología asistencial que cada uno requiera (v.g. sillas de ruedas).

De este modo, mientras la Tecnología de la Rehabilitación orienta su trabajo a una población restringida, el Diseño para Todos debería ser aplicado por todos los diseñadores a todos los productos. Por ello es importante que en la formación de los

diseñadores se incluya esta disciplina [9]. Para ello, basta con dotar al diseñador con: (1) pautas de diseño incluyentes [10, 11]; (2) metodologías centradas en el usuario, la tarea y el contexto (v.g. USERfit [12] desarrollada dentro del proyecto europeo USER) y (3) herramientas de diseño adecuadas (v.g. USERfit Tool [13], desarrollada por el Laboratorio de Interacción Persona-Computador para Necesidades Especiales –LIPCNE– para facilitar la aplicación de la metodología USERfit).

9.1.2 Legislación sobre accesibilidad a la web

Dependiendo del país y de la fuente de información, el porcentaje de personas con discapacidad ronda entre el 10 y el 20% de la población. Además, aunque las discapacidades congénitas están disminuyendo en los países desarrollados, se espera un aumento de estas cifras debido al envejecimiento de la población y a los accidentes de tráfico. Por ello, se están realizando esfuerzos con el fin de crear un entorno favorable para que la industria y las organizaciones incorporen la accesibilidad entre sus objetivos. El propósito es pasar de un enfoque reactivo a la accesibilidad, en el que se fomenta la adaptación de productos y servicios a las capacidades de los usuarios, a un enfoque proactivo, donde se fomente el desarrollo de productos accesibles en todas sus fases: concepción, diseño e implementación.

Para ello se trata de introducir normas a diferentes niveles: **`legislación`** que obligue a diseñar servicios y sistemas accesibles; **`estándares y normas`**, que permitan especificar los requisitos mínimos de los servicios accesibles, y **`pautas`** que ayuden a los diseñadores con poca experiencia en este ámbito a resolver los problemas de accesibilidad sin necesidad de realizar complejos test de usuario. Este apartado se centra en la legislación y la estandarización. En el apartado 9.1.3 se presentan las pautas más referenciadas en la accesibilidad a la web.

9.1.2.1 Legislación El objetivo actual es fomentar desarrollos tecnológicos, en este caso aplicados a la web, dirigidos a promocionar el diseño para todos, lo que implica considerar la accesibilidad desde la fase inicial de diseño (de acuerdo a la aproximación proactiva), proveer adaptaciones o alternativas como opciones estándar, añadir mecanismos asistenciales especiales a productos existentes y personalizar los productos. En los siguientes párrafos se mencionan algunas de las leyes que han tenido más impacto en lo que respecta a la accesibilidad.

Americans with Disabilities Act (ADA) fue promulgada en 1990 y ha sido precursora en aspectos de accesibilidad. Esta ley impide al gobierno USA apoyar programas y adquirir equipamiento que discriminen a las personas con discapacidades, con mención explícita a la accesibilidad web [14]. A pesar de ser una ley nacional, tiene un efecto importante sobre el resto de los países que adquieren equipamiento producido por o para compañías norteamericanas.

Rehabilitation Act, Section 508 fue promulgada en 1986, aunque posteriormente ha sido enmendada en 1992 y 1998. Exige que los servicios electrónicos ofrecidos por el gobierno sean igualmente accesibles para empleados con y sin discapacidades. Dicha responsabilidad recae tanto sobre la Agencia Federal como sobre la industria, responsable de buscar soluciones accesibles cuando esté involucrada [15]. La *Section*

508 también incluye pautas que suelen utilizarse de referente en la accesibilidad a la web en varias de las herramientas que se presentan en la sección 9.3.

La situación legal en Europa es bastante variada. La Unión Europea, a través de la iniciativa *eAccessibility* integrada en la acción *eEurope* promovió la adopción, en diciembre de 1999, de la iniciativa "*eEurope – Una sociedad de la información para todos*" [16] que pretende superar los obstáculos que actualmente limitan la asimilación de las tecnologías digitales en Europa, a través de la consecución de tres objetivos clave, como son conseguir que todos los ciudadanos, hogares, escuelas, empresas y administraciones estén conectadas a la red, crear en Europa una cultura y un espíritu empresarial abierto a la cultura digital, y garantizar que la sociedad de la información no se traduzca en exclusión social. Por su parte, en 2002 el Parlamento Europeo aprobó el documento "*eEurope 2002, Accessibility of public websites and their content*" [17] acordado por la Comisión Europea y todos los estados miembros. En julio de 2002 se estableció la red *EdeAN (European Design for All e-Accessibility Network*) de acuerdo a objetivos específicos del *eEurope 2002 Action Plan*.

En lo que respecta al Estado Español, la ley 51/2003, de 2 de diciembre, de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad menciona en su disposición final séptima las "*Condiciones básicas de accesibilidad y no discriminación para el acceso y utilización de las tecnologías, productos y servicios relacionados con la sociedad de la información y medios de comunicación social*". Entre ellas, destaca que:

"*En el plazo de dos años desde la entrada en vigor de esta ley, el Gobierno aprobará unas condiciones básicas de accesibilidad y no discriminación para el acceso y utilización de las tecnologías, productos y servicios relacionados con la sociedad de la información y de cualquier medio de comunicación social, que serán obligatorios en el plazo de cuatro a seis años"*. Asimismo, *"en el plazo de dos años desde la entrada en vigor de esta ley, el Gobierno deberá realizar los estudios integrales sobre la accesibilidad a dichos bienes o servicios que se consideren más relevantes desde el punto de vista de la no discriminación y accesibilidad universal*".

Por su parte, la ley 34/2002 de 11 de julio de servicios de la sociedad de la información y de comercio electrónico (LSSICE) menciona en su disposición quinta cuestiones relativas a la "*accesibilidad para las personas con discapacidad y de edad avanzada a la información proporcionada por medios electrónicos*". Señala que:

"*Las administraciones públicas adoptarán las medidas necesarias para que la información disponible en sus respectivas páginas de Internet pueda ser accesible a personas con discapacidad y de edad avanzada de acuerdo con los criterios de accesibilidad al contenido generalmente reconocidos antes del 31 de diciembre de 2005. Asimismo, podrán exigir que las páginas de Internet cuyo diseño o mantenimiento financien apliquen los criterios de accesibilidad antes mencionados. Igualmente, se promoverá la adopción de normas de accesibilidad por los prestadores de servicios y los fabricantes de equipos y software, para facilitar el acceso de las personas con discapacidad o de edad avanzada a los contenidos digitales*".

9.1.2.2 Estandarización. La estandarización especifica en detalle las características de los productos para que sean compatibles entre sí y con otras tecnologías. Usualmente la ley se basa en la aplicación de estándares previamente

definidos, ya que se puede verificar su cumplimiento de manera no ambigua. Sin embargo, los rápidos avances en tecnología web han impedido la creación de estándares *de iure*, desarrollados por organismos oficiales (v.g. *International Organization for Standarization* – ISO), o por otros organismos (estandarización *ad hoc*). Solamente algunos estándares *de facto*, impuestos por la industria (v.g. *IBM web page and Java accessibility*, *Microsoft web accessibility*, *Sun Java accessibility*), han sido posibles.

Sin embargo existe un estándar oficial dirigido a la accesibilidad de sistemas interactivos que incluye soluciones para aplicaciones ofimáticas, páginas web y multimedia. Asimismo, promueve la usabilidad de sistemas en combinación de tecnología asistencial cuando se requiera. El ISO/TS 16071:2003 "*Ergonomics of human-system interaction – Guidance on accessibility for human-computer interfaces*" proporciona guías de diseño de *software* accesible para entornos laborales, educativos y del hogar para personas con un amplio rango de discapacidades (visuales, auditivas, motoras, cognitivas, etc.).

9.1.3 Pautas WAI de accesibilidad

El Consorcio de la *World Wide Web* (W3C o *World Wide Web Consortium*) surgió a mediados de los años 90 formado por miembros del ámbito industrial, académico y de la administración. El W3C fue creado para lograr todo el potencial de la web desarrollando protocolos que promuevan su evolución y aseguren su interoperabilidad [18]. Los servicios del W3C incluyen:

- un repositorio de información sobre la web que pueda ser consultado por desarrolladores y usuarios,
- código de referencia a modo de ejemplo para promover los estándares, y
- ejemplos de prototipos y aplicaciones para demostrar el uso de las nuevas tecnologías.

Sus actividades están desglosadas en varios grupos de trabajo. Uno de ellos es la **Iniciativa de Accesibilidad a la Web** (WAI: *Web Accessibility Initiative*)

9.1.3.1 Iniciativa de accesibilidad a la web. Surgida en 1997, este grupo de trabajo del consorcio W3C trata la accesibilidad a la web a través de las siguientes actividades complementarias [19]:

- asegurar que la tecnología de la web tiene en cuenta la accesibilidad,
- desarrollar pautas de accesibilidad,
- desarrollar herramientas que faciliten la evaluación y reparación de los sitios web,
- dirigir la educación y divulgación, y
- coordinar con los esfuerzos de investigación y desarrollo que puedan tener que ver con la accesibilidad a la web.

La WAI coopera con organizaciones de todo el mundo incluyendo organizaciones de personas con discapacidades y de investigación sobre accesibilidad, industria y gobiernos interesados en promover la creación de contenido web accesible. Entre los patrocinadores de la WAI se encuentran el Instituto Nacional sobre la Investigación

en Discapacidad y Rehabilitación del Departamento de Educación de Estados Unidos, el departamento de Gobierno e Industria de Canadá, el Programa sobre las Tecnologías de la Sociedad de la Información de la Comisión Europea, la Fundación ONCE, HP, IBM, la Corporación Microsoft, etc.

En los siguientes apartados se describen en mayor detalle los trabajos desarrollados en la WAI dentro de las actividades relativas al desarrollo de pautas de accesibilidad, tanto en lo relativo al contenido web, como a las herramientas de autor y los agentes de usuario.

9.1.3.2 ***Web Content Accessibility Guidelines*** **(WCAG).** Las **pautas de accesibilidad al contenido web** son recomendaciones para que las páginas web sean accesibles para todos con ayuda de la tecnología estándar existente. Estas pautas, sin carácter regulador, pueden ser adoptadas formal o informalmente por diversos tipos de organizaciones para establecer el nivel de accesibilidad que esas organizaciones vayan a cumplir en todas o varias de sus páginas web [20].

La versión 1.0 de dichas pautas fue establecida el 5 de mayo de 1999 y es uno de los referentes principales a la hora de analizar la accesibilidad de los contenidos en la web. En dicha versión se contemplan 14 pautas genéricas compuestas de varios puntos de verificación que pueden tener tres niveles de prioridad posibles.

La prioridad 1 se refiere a los puntos de verificación que tienen que satisfacer los desarrolladores de contenidos para que no existan usuarios que no sean capaces de acceder a la información de un sitio web. La prioridad 2 es para los puntos que un desarrollador debería satisfacer para evitar que la información sea difícilmente accesible. La prioridad 3 es para los puntos que pueden satisfacerse, si se quiere evitar que determinados usuarios puedan encontrar dificultades al acceder a la información.

Se asignan **niveles de conformidad** a los sitios web en función de los puntos de verificación que se satisfagan y sus niveles de prioridad. Si se cumplen todos los puntos de prioridad 1, la conformidad corresponde al nivel A; si se cumplen los de prioridad 1 y 2, le corresponde el nivel de conformidad de nivel Doble-A; por fin, si se cumplen todos los puntos de prioridad 1, 2 y 3, el nivel de conformidad conseguido es el Triple-A. Estos niveles se facilitan para que sean referenciados por otras organizaciones y la WAI proporciona logotipos para indicar los niveles de conformidad logrados por los sitios web.

El formato de las 14 pautas de la versión 1.0 incluye una breve descripción, los puntos de verificación con sus correspondientes prioridades, algunos ejemplos, y las técnicas asociadas para conseguir su cumplimiento.

Sin embargo, y debido a la evolución de la tecnología, se está trabajando desde hace tiempo en la versión 2.0 de las pautas de accesibilidad. La última versión publicada hasta la fecha es un borrador de trabajo (que todavía no es definitivo) en donde se plantean cuatro principios y tres niveles de conformidad. En concreto, los principios se refieren a que el contenido de una página debe ser **perceptible** por cualquier usuario, los elementos de la interfaz deben ser **comprensibles** para todos, el contenido y los controles deben ser fácilmente **manejables** y el diseño debe ser **robusto** para poder ser procesado por las tecnologías actuales y futuras.

En cuanto a los niveles de conformidad, el nivel 1 busca alcanzar un nivel mínimo de accesibilidad mediante etiquetado, *scripting* u otras tecnologías. El nivel 2 busca incrementar la accesibilidad mediante contenidos y presentaciones directamente

accesibles. Por su lado, el nivel 3 busca incrementar tanto la accesibilidad directa como la del agente de usuario.

Las principales diferencias entre la versión 2.0 y la 1.0 son: se eliminan las largas listas sobre lo que se debe y no se debe hacer, se concentra en los principios básicos de la accesibilidad y las pautas referentes a ellos, y se evita discutir acerca de cualquier tecnología específica.

9.1.3.3 ***User Agent Accessibility Guidelines*** **(UAAG).** Las **`pautas de accesibilidad a los agentes de usuario`** son recomendaciones para que los navegadores y programas multimedia sean accesibles para todos y para que estas herramientas puedan cooperar mejor con los dispositivos de tecnología asistencial [21].

La versión 1.0 establecida en diciembre de 2002 se compone de 12 pautas dotadas de puntos de verificación que cuentan con tres niveles de prioridad análogos a los de WCAG 1.0. Los niveles de conformidad que se pueden conseguir son también A, Doble-A y Triple-A, de manera análoga a lo que ocurre en las WCAG 1.0.

9.1.3.4 ***Authoring Tool Accessibility Guidelines*** **(ATAG).** Las **`pautas de accesibilidad a las herramientas de autor`** son recomendaciones para que las herramientas de diseño de páginas web sean accesibles para todos, así como el resultado generado por ellas [22].

La versión 1.0 establecida en febrero de 2000 se compone de 7 pautas compuestas por puntos de verificación con tres niveles de prioridad similares a los indicados anteriormente.

Como ocurre con las WCAG, actualmente se está trabajando en la versión 2.0 de las ATAG que consiste en un borrador de trabajo compuesto por 4 pautas y tres niveles de conformidad.

9.1.4 Ventajas inherentes al diseño de sitios accesibles

Muchos diseñadores de páginas web pueden sentir inicialmente rechazo hacia el desarrollo de páginas web accesibles. Esto puede deberse a los muchos tópicos acerca de la complejidad añadida por la accesibilidad. Se dice que es un proceso caro, que coarta la inspiración personal del diseñador, que existe poca información, etc. En este apartado se discuten esos tópicos y se exponen explícitamente algunas ventajas asociadas al diseño de páginas accesibles [23].

Las páginas accesibles pueden ser diseñadas tan creativamente como las inaccesibles. El objetivo es asegurar que todos los tipos de sitios web trabajen bien con cualquier tipo de usuarios. Las páginas web accesibles han de diseñarse con la suficiente flexibilidad para que puedan ser operadas de maneras diferentes (con el ratón o el teclado) y vistas con diferentes navegadores y/o pantallas. Una página web compuesta únicamente por texto no tiene por qué ser necesariamente accesible, ni las páginas accesibles tienen por qué estar realizadas con sólo texto.

Diseñar una página accesible desde el principio no supone un incremento de los costes de desarrollo. De hecho, algunos aspectos tales como las hojas de estilo pueden minimizar los costes de mantenimiento y actualización de las páginas web. Esta

tendencia es previsible que vaya en aumento porque el soporte que dan las herramientas de autor y los navegadores a las hojas de estilo va mejorando con el tiempo.

Adaptar los sitios web existentes a la accesibilidad depende de factores tales como el tamaño del sitio, su complejidad y la herramienta de autor que se utilice para su edición y actualización. Las actualizaciones y revisiones periódicas de los sitios son buenas oportunidades para revisar su accesibilidad y analizar si los cambios introducidos merecen la pena desde el punto de vista de coste/eficiencia, y si se ha llegado a una audiencia mayor y con un grado mayor de usabilidad.

Entre las ventajas adicionales que se consiguen al hacer accesibles las páginas, es necesario mencionar que aumenta la cuota de mercado y la audiencia a la que se llega [24]. Ello es debido, principalmente, a que la población con algún tipo de discapacidad llega en algunos países a ser cercana al 20%. Además el diseño de páginas accesibles mejora de la usabilidad y facilita el acceso a las personas con problemas de lectura (incluyendo a las personas cuyo idioma materno no es el de las páginas web que visitan). Por otro lado, mejora la visibilidad del sitio para los buscadores web, facilita el soporte a la web semántica, la adaptación del contenido a múltiples formatos y dispositivos (al separar la estructura y la presentación), mejora la internacionalización y facilita el acceso a usuarios con conexiones de bajo ancho de banda. La mejora en la eficiencia revierte en la facilidad de mantenimiento de los sitios y la disminución de la carga que soportan los servidores de direcciones, y la optimización del uso de su ancho de banda.

Por otro lado, las organizaciones que realizan páginas accesibles demuestran una responsabilidad social que puede aportarles propaganda favorable y las responsabilidades legales que puedan plantearse por no hacer páginas accesibles serán menores o inexistentes.

El argumento de que la información sobre la accesibilidad es escasa, hoy en día no se mantiene ya que existen multitud de publicaciones y sitios web (principalmente el propio de la WAI [19]) en donde hay abundantes información y recursos sobre la accesibilidad a la web disponibles. No obstante, como se ha mencionado previamente, es necesario un esfuerzo para introducir en la formación de los diseñadores la filosofía del Diseño para Todos, el uso de pautas de diseño y la práctica con metodologías y herramientas adecuadas.

9.1.5 Retos futuros

En la actualidad, a pesar del marco legal vigente (v.g. ley 51/2003), el número de organismos públicos cuyas páginas web son accesibles sigue siendo muy limitado. Sirva como ejemplo el informe en que se detalla el estado actual de accesibilidad de los portales de las universidades públicas españolas [25]. Por lo tanto, es primordial seguir ahondando en la divulgación de la accesibilidad para que ésta sea tenida en cuenta en el diseño web.

Respecto de la tecnología, uno de los retos con los que se enfrenta la accesibilidad web tiene que ver con las páginas generadas de manera dinámica, usualmente a partir de bases de datos que almacenan segmentos de los contenidos que serán combinados para crear las páginas web a visualizar. En estos casos, hay que almacenar toda la

información necesaria en los repositorios del sistema para que, a partir de los datos y con los modelos de las páginas a mostrar (v.g. *templates*), las páginas y servicios resultantes sean accesibles, tal como se verá en el apartado 9.4.2.1.

Respecto al uso de soluciones específicas de una marca a la hora de realizar un diseño Web, ninguna otra ha tenido tanto éxito como *Flash*, que permite embeber animaciones y gráficos vectoriales. Aparte de sus capacidades propias, su éxito radica en gran medida en que funciona igualmente en un gran número de navegadores sin los problemas existentes con otras tecnologías específicas o estándares. Sin embargo, a pesar de que en los últimos tiempos ha habido mejoras al respecto [26], esta tecnología aún presenta dificultades para crear contenidos accesibles. Dada la resistencia de los diseñadores a utilizar esta tecnología siguiendo las recomendaciones de accesibilidad, ocurre que muchas de las animaciones producidas sólo funcionan en navegadores concretos y son completamente inaccesibles para dispositivos y navegadores sin soporte a *Flash*. Por tanto, uno de los retos actuales es conseguir que el código específico de cada marca soporte las características de accesibilidad.

La generación de código accesible por parte de las herramientas de autor utilizadas para crear páginas web también es un aspecto a tener en cuenta. Hasta fechas recientes, el desarrollo de código mediante estándares y el soporte de la accesibilidad había sido una preocupación secundaria en una situación en la que los navegadores no cumplían con los estándares web, como se explica en la sección 9.2. Como se ha mencionado anteriormente, la iniciativa WAI ha creado las ATAG, pautas para las herramientas de autor que generan código. La evolución de los navegadores hacia el correcto uso de los estándares y una mayor concienciación de la importancia de la accesibilidad en el diseño web han propiciado un cambio de actitud. En la actualidad, se están haciendo esfuerzos por parte de las compañías desarrolladoras de herramientas de autor para el diseño web con el fin de cumplir con las ATAG [27]. Sin embargo, para poder generar código accesible es necesario, además del uso de herramientas apropiadas, un correcto conocimiento y uso de los estándares web.

9.2 Diseño accesible. Separación entre contenido, presentación y comportamiento

El acceso universal a la información ha sido uno de los objetivos de la web desde sus orígenes y uno de los pilares básicos en la consecución de dicho objetivo es el desarrollo por parte del consorcio W3C de tecnologías para creación e interpretación de contenido, conocidas como estándares web.

Los estándares web son diseñados cuidadosamente para conseguir el mayor beneficio posible para el mayor número de usuarios de la web y para asegurar la viabilidad a largo plazo de cualquier documento publicado en ella. Aunque históricamente ha habido serias dificultades para la adopción de estos estándares, en la actualidad el desarrollo de lenguajes de marcado extensibles ha supuesto un gran impulso para su uso.

Concretamente, los estándares propuestos pretenden evitar una de las causas reconocidas de los problemas de accesibilidad a la web que es el hecho de que, en

muchos casos, el contenido, la presentación y el comportamiento de la información se codifiquen juntos.

A continuación se describe el desarrollo de los estándares web, así como aquellos propuestos para realizar diseños accesibles.

9.2.1 Introducción: desarrollo histórico de los estándares web

La tecnología en la que se basa la web (el *HyperText Transfer Protocol*, HTTP) fue presentada originalmente en 1990 como soporte para un sistema de intercambio de información entre científicos. Los elementos necesarios para su desarrollo incluían la especificación de un lenguaje de marcado de documentos basado en hipertexto, al que se dio el nombre de HTML (*HyperText Markup Language*), y que constituye la base del diseño web [28].

Para poder acceder a documentos, usualmente alojados en servidores, compuestos siguiendo el formato HTML era necesario disponer de navegadores, es decir de programas de visualización del contenido que permitieran avanzar a través de la estructura hipertexto. En poco tiempo, el desarrollo de navegadores y el paulatino aumento de servidores web que alojaban documentos de hipertexto, popularizó la web de manera que otros tipos de usuarios empezaron a interesarse por ella.

El nuevo espectro de usuarios aportó una serie de necesidades, principalmente en lo referente a la visualización de los documentos web. HTML resultó ser un formato demasiado restrictivo para mostrar ciertos contenidos. Por ello, los fabricantes empezaron a crear etiquetas propias (no incluidas en el HTML "estándar") con las que dar respuesta a dichas necesidades. Por otro lado, se diseñaron navegadores orientados a trabajar con las tecnologías específicas de cada fabricante más que al cumplimiento de la especificación estándar. Al surgir problemas de compatibilidad de los documentos en diferentes plataformas y navegadores, los diseñadores comenzaron a crear contenidos usando marcado no estándar para que pudieran ser visibles con diversos navegadores y a utilizar tecnologías propias de navegadores concretos.

A partir de 1995 comenzó la denominada "*guerra de los navegadores*". *Netscape* e *Internet Explorer*, que eran los dominantes, se enzarzaron en una batalla comercial por conseguir cuota de mercado que finalmente se decantó del lado del *Internet Explorer*. Como consecuencia, los estándares web quedaron seriamente debilitados por cuanto gran parte de la comunidad diseñadora comenzó a realizar documentos para ser visualizados preferentemente mediante el navegador más extendido, utilizando para ello las tecnologías específicas que soportaba este navegador, en vez de utilizar los estándares, que no cumplía completamente.

Desde la creación del consorcio W3C [18] en 1994, el uso de estándares empezó a tomar un nuevo impulso. La adopción del XML (*eXtensible Markup Language*) en 1998 revolucionó el mundo de la informática, ya que era la primera vez que se disponía de un formato adaptable y universal para estructurar documentos e información. XML solucionó los problemas de incompatibilidad entre formatos específicos existentes hasta la fecha y permitió el desarrollo de nuevos estándares mediante la capacidad de creación de nuevas etiquetas para definir nuevos lenguajes.

En lo que a la web se refiere, la adopción de XML como estándar web, junto con el hecho de que en el consorcio W3C estuvieran incluidos los principales

desarrolladores de navegadores, se tradujo en la progresiva popularización del uso lo que se ha dado en llamar la *trinidad de los estándares web* [1], que se puede ver en la Figura 9.1.

Fig. 9.1 Separación del contenido, la presentación y el comportamiento.

El objetivo es que el uso de los estándares web solucione los problemas de diseño dividiendo las páginas en tres componentes separados: **`contenido`**, **`presentación`** y **`comportamiento`**. Esta división racionaliza el desarrollo web de manera que, dependiendo de los objetivos y la audiencia, los diseñadores y desarrolladores pueden optar por aplicar diferentes tipos de desarrollos y diseños.

En sitios web muy visitados por navegadores antiguos, suele ser recomendable utilizar un **`diseño transicional`**, ya que de esta manera se mantiene una razonable compatibilidad hacia atrás utilizando estándares web, plantea menos problemas de mantenimiento al eliminar código específico y ofrece la oportunidad de incrementar la accesibilidad, además de permitir la compatibilidad hacia adelante (por el uso de estándares web). La compatibilidad hacia atrás supone la compatibilidad con versiones previas de los navegadores y dispositivos, mientras que la compatibilidad hacia adelante consiste en la compatibilidad con versiones actuales y futuras de los navegadores y dispositivos.

Aplicando estrictamente el criterio de separación total entre contenido, presentación y comportamiento se realiza un **`diseño estricto`**, que proporciona compatibilidad hacia adelante, ya que permite una mucho mayor interoperabilidad con navegadores y dispositivos existentes y futuros. Al estar separado el contenido de la presentación, se consigue llegar a más usuarios con menos trabajo, ya que desaparece la necesidad de crear diferentes versiones de documentos. Ello también redunda en beneficio de la accesibilidad, ya que, como se ha dicho, un correcto uso de los estándares web implica una gran reducción en el número de problemas de accesibilidad, cuando no su total desaparición.

A continuación se explican con más detalle los estándares concretos definidos para el contenido, la presentación y el comportamiento.

9.2.2 Contenido

El lenguaje HTML, ampliamente considerado como lenguaje de publicación estándar de la web, se sirve de un reducido conjunto de etiquetas estructurales y semánticas apropiadas para la realización de documentos relativamente simples [28]. Contiene datos en forma de texto formateados de acuerdo a su significado estructural: cabecera, cabecera secundaria, párrafo, lista numerada, lista de definición, etc. Asimismo, contiene estructuras adicionales consideradas necesarias por el diseñador, así como objetos embebidos tales como imágenes, animaciones *Flash*, objetos *Quicktime*, etc. y sus equivalencias para aquellos casos en los que no se pueda acceder a ellos.

Como se ha mencionado en el anterior apartado, la constante invención de nuevos elementos para ser usados dentro de HTML acarreó graves problemas de

compatibilidad de documentos entre las diferentes plataformas y navegadores. El consorcio W3C solucionó el problema del soporte en navegadores combinando la familiar simplicidad de HTML con el poder de crear lenguajes extensibles de XML, dando lugar así al lenguaje de marcado estándar XHTML (*eXtensible HyperText Markup Language*).

9.2.2.1 XHTML. El consorcio W3C define el XHTML como una familia de módulos y documentos presentes y futuros que reproduce, engloba y extiende HTML, reformulado en XML [29]. Todas las familias de tipos de documentos están basadas en XML y están diseñadas para trabajar con agentes de usuario basados en XML.

El grupo de trabajo de XHTML ha ido desarrollando una serie de especificaciones para este sucesor del HTML. XHTML 1.0 [30] es la base del formato, en la que se especifica la sintaxis y la definición normativa de documentos conformes al estándar. XHTML 1.0 permite la descomposición en módulos resumen [31] que proporcionan tipos de funcionalidad específicos, ya que no todos los elementos XHTML son necesarios en todas las plataformas. Con posterioridad fue desarrollado XHTML 1.1 [32], en el que se establece la base para futuros tipos de documentos extendidos de la familia XHTML mediante su reformulación mediante módulos. En la actualidad se está desarrollando la versión 2.0 del estándar [33]

XHTML proporciona una serie de ventajas desde el punto de vista de la accesibilidad:

- al estar basado en XML, puede trabajar correctamente con otros lenguajes de marcado, aplicaciones y protocolos basados en XML, tales como MathML y SVG (*Scalable Vector Graphics*), permitiendo también una mejor transformación del contenido,
- el soporte a XHTML está garantizado en las más modernas versiones de navegadores, mientras que en navegadores antiguos también puede ser usado,
- manejado de forma correcta, es completamente portable a navegadores web, lectores de pantalla, navegadores de texto, dispositivos móviles, y otros agentes de usuario especializados,
- su utilización permite romper el hábito de utilizar etiquetas de marcado para ajustar la presentación,
- ofrece la posibilidad de testar el código utilizando validadores [34], lo que evita el esfuerzo de la revisión manual y ayuda a evitar muchos errores de accesibilidad básicos, y
- XHTML 1.0 es un puente hacia futuras versiones de HTML, ya que su adaptación hacia ellas será mucho más sencilla que desde HTML

9.2.3 Presentación

La separación de la estructura del documento de su presentación ha sido un objetivo de HTML desde su creación. En un principio se consideró que era asunto de cada navegador decidir cómo presentar de la mejor manera las páginas a sus usuarios, lo que condujo a la proliferación de diferentes maneras de visualización del contenido.

Cuando se estableció el consorcio W3C, éste decidió abordar el problema mediante el uso de **`hojas de estilo`**, especificadas mediante el estándar CSS.

9.2.3.1 Hojas de estilo. *Cascading Style Sheet* (CSS) es una especificación para la presentación de documentos HTML [35]. Define un lenguaje para la creación de hojas de estilo, que funcionan como una plantilla, permitiendo a los desarrolladores y diseñadores web definir un estilo para elementos HTML y aplicarlos a tantos elementos y documentos como deseen.

La primera especificación del estándar fue CSS1 [36], en la que fueron definidos los aspectos básicos de las hojas de estilo, tales como la herencia de estilos, pseudo-clases y pseudo-elementos, modelo de formato y propiedades de estilo para elementos de hipertexto como fuentes, color, fondo y texto. Posteriormente se ha desarrollado una segunda especificación del estándar [37], bastante más voluminosa que la primera, en la que se profundiza en las características definidas en CSS1 y se definen nuevas funcionalidades, tales como el concepto de tipo de medio, hojas de estilo auditivas y soporte de estilo para tablas. En estos momentos se está desarrollando la tercera versión del estándar [38] con la particularidad de que será una especificación modularizada en la que está previsto ir desarrollando diferentes módulos referentes a aspectos concretos, como por ejemplo los módulos de tablas, *math* y SVG.

CSS proporciona una serie de ventajas de cara a la mejora de la accesibilidad que se exponen a continuación [39]:

- separa la estructura del documento de la presentación,
- permite que los usuarios eliminen estilos del autor si lo desean,
- si la hoja de estilo del usuario contiene el operador `!important`, tiene prioridad sobre cualquier atributo aplicable en una hoja de estilo del autor,
- proporciona control exacto sobre aspectos como el tamaño de la fuente, el color, y el estilo,
- utiliza fuentes y colores del sistema para que las páginas se adapten a las necesidades y preferencias del usuario,
- usa unidades relativas en lugar de unidades absolutas en los atributos del lenguaje,
- controla la representación de los textos,
- controla la composición y alineación de los elementos, y
- permite la herencia de estilos

9.2.4 Comportamiento

Mediante el uso de un modelo de objetos estándar y un lenguaje estándar para manipular dicho modelo se pueden crear comportamientos sofisticados y efectos que funcionen en múltiples plataformas y navegadores sobre los estándares de contenido y presentación anteriormente descritos. El W3C propone el uso del **`Modelo de Objetos del Documento`** (*Document Object Model o* DOM) como modelo de objetos estándar, mientras que para manipular DOM la recomendación es el uso de un nuevo lenguaje de *script* estándar llamado *ECMAScript*.

9.2.4.1 DOM. DOM fue definido por el W3C como una interfaz independiente de navegador, plataforma y lenguaje que permite a los programas y *scripts* acceder y actualizar dinámicamente el contenido, la estructura y el estilo de documentos [40]. De esta manera, dicho documento puede ser procesado y el resultado de dicho procesamiento posteriormente incorporado al documento. Define la estructura lógica de los documentos y la manera en que un documento es accedido y manipulado.

En la primera versión, DOM nivel 1 [41], se ofrece soporte para XML y HTML, mientras que en la segunda versión, DOM nivel 2 [42], el soporte se extiende a XML con espacios nominales y también a CSS, lo que resulta de gran importancia desde el punto de vista de la accesibilidad. En la última versión de DOM existente hasta la fecha, DOM nivel 3 [43], se amplía el soporte XML y se incluye soporte para validación, capacidad de cargar y salvar documentos, vocabularios de marcado combinados y sus implicaciones con la interfaz de programación de aplicaciones DOM (DOM embebido).

9.2.4.2 Lenguajes de *script*. Al igual que con el estilo, los primeros navegadores tampoco tenían una manera común de facilitar comportamientos sofisticados mediante *scripts*. Cada navegador tenía su propio modelo de objetos sobre el que trabajar y además tampoco existía un lenguaje *script* estándar. *Netscape* desarrolló *JavaScript*, mientras que *Microsoft* creó *JScript* a la vez que desarrollaba la tecnología específica *ActiveX*. La situación vivida con los lenguajes *script* fue exactamente la misma que la vivida durante la "guerra" de los navegadores: los desarrolladores y diseñadores no disponían de un lenguaje *script* estándar sobre el que crear comportamientos sofisticados.

ECMAScript es un lenguaje *script* estándar desarrollado por la asociación europea de fabricantes de ordenadores (ECMA - *European Computer Manufacturers Association*) [44]. Este lenguaje manipula objetos provenientes de documentos web especificados usando DOM, que pueden ser añadidos, borrados o cambiados. Al ser un lenguaje estándar, este lenguaje es soportado por las versiones más modernas de los navegadores. El estándar *ECMAScript* está basado en *JavaScript* y *JScript*. Las últimas especificaciones de *ECMAScript* están disponibles en [45].

9.3 Herramientas avanzadas de accesibilidad

La accesibilidad a la web es un concepto amplio cuya aplicación exige a los diseñadores tener en cuenta numerosos detalles. Con el ánimo de facilitar la aplicación de todos los requisitos, diferentes entidades (compañías, grupos de investigación, asociaciones de usuarios, etc.) han desarrollado herramientas para tratar la accesibilidad. Esas herramientas pueden clasificarse atendiendo a varios conceptos [46]:

- **`generales`** o **`específicas`**, dependiendo de si contemplan la accesibilidad en todo su conjunto o sólo una parte de ella,
- ***`online`*** (las que se ejecutan en un nodo remoto y ofrecen sus servicios a través de la red) o en el **`puesto de trabajo`** (las que se ejecutan localmente en el propio ordenador del usuario),

- las que tratan **una sola página** o **todo un sitio web** (v.g. *crawlers*),
- **programas independientes** (*stand-alone*) o a modo de **añadido en otros programas** (*plug-ins*),
- de **validación** (que testan las páginas de acuerdo a una serie de reglas), de **reparación** (que sugieren como reparar errores de accesibilidad) o de **transformación** (que modifican la presentación de acuerdo a determinados requisitos).

Además, cada herramienta presenta diversas características, tanto funcionales como de usabilidad.

En los siguientes apartados se presentan algunas herramientas de validación y reparación. Las de transformación son habitualmente herramientas específicas que detectan problemas concretos y resultan de gran ayuda en el prototipado para el desarrollo de sitios web. A menudo, los resultados de las herramientas de transformación pueden aplicarse también en aspectos relativos a la usabilidad.

De cualquier manera, es recomendable el uso de diversas herramientas a lo largo de las fases de diseño, implementación y operación de las páginas web. No conviene confiar en los resultados de una sola herramienta, sobre todo teniendo en cuenta que el objetivo final es implementar páginas web accesibles en lugar de satisfacer los requisitos de una herramienta concreta [46].

9.3.1 Herramientas de validación automática de la accesibilidad

Estas herramientas analizan el cumplimiento de un conjunto de pautas de accesibilidad (usualmente las WCAG o *Section 508*). Actualmente, la mayoría de ellas no están incluidas dentro de herramientas de diseño web lo que dificulta su uso durante el transcurso del diseño. Entre las herramientas de validación generales, se pueden destacar las siguientes:

- *Bobby™* [47]: desarrollada por CAST. Existen dos versiones, una gratuita *online* y otra de pago para su ejecución local.
- *Wave* [48]: desarrollada por *WEB Accessibility In Mind* (Webaim) y *Pennsylvania's Initiative on Assistive Technology* (PIAT).
- *Test de Accesibilidad a la Web (TAW)* [49]. Desarrollada por la Fundación CTIC (Centro Tecnológico de la Información y la ComunicacIón) y financiada por el Centro Estatal de Autonomía Personal y Ayudas Técnicas (CEAPAT), evalúa una página basándose en las pautas WCAG.
- *EvalAccess* [50]: Herramienta de validación *online* desarrollada por el Laboratorio de Interacción Persona-Computador para Necesidades Especiales (UPV/EHU). Se habla en mayor profundidad de esta herramienta en el apartado 9.3.3.

Además existen herramientas específicas para verificar si el código HTML, XHTML o CSS cumple el estándar correspondiente, lo que permite detectar elementos de código que han pasado a estar en desuso, así como el código basado en tecnologías específicas. Entre las herramientas de validación específicas se pueden destacar las siguientes:

- *W3C CSS Validator* [51]: Valida el código CSS usado en los documentos.

- *W3C HTML Validator Service* [34]: Valida el código HTML basándose en las recomendaciones W3C y el estándar HTML.
- *WDG HTML Validator* [52]: Está basado en el mismo motor que el *W3C HTML Validator Service* y puede usarse tanto *online* como localmente.

9.3.2 Herramientas de ayuda a la corrección

A continuación se mencionan algunas de las herramientas de ayuda a la corrección generales [53].

- *Tidy* [54]: Sirve para reparar errores de HTML.
- *AccRepair* [55]: Verifica y corrige la accesibilidad de los sitios web basándose en las pautas WCAG y en las de la Section 508.
- *A-Prompt* [56]: Desarrollado por la Universidad de Toronto (Canadá), identifica los problemas de accesibilidad y ayuda a corregirlos.

9.3.3 Herramienta de validación *EvalAccess*

Dado el interés de la comunidad científica en la mejora de la accesibilidad web de las personas con discapacidad, las pautas de accesibilidad son continuamente analizadas y mejoradas. De este modo, van apareciendo nuevas versiones de los conjuntos de pautas que las herramientas de validación tienen que verificar. Las herramientas que han sido diseñadas específicamente para un determinado conjunto de pautas tienen que ser modificadas cada vez que aparece una nueva versión de las mismas. Para evitar este problema es necesario crear un nuevo tipo de herramienta de validación que pueda llevar a cabo este proceso independientemente del conjunto de pautas de accesibilidad a evaluar. Esta es la filosofía de desarrollo de herramientas tales como *Kwaresmi* [57] y *EvalAccess* [58]. Esta última se describe en los siguientes apartados.

9.3.3.1 Características de EvalAccess. La herramienta *EvalAccess* presenta las siguientes características:

- Es de tipo general, es decir, valida múltiples aspectos de la accesibilidad.
- Es gratuita, todo usuario puede acceder a la herramienta y validar cualquier página web.
- Es una herramienta *online*.
- Valida tanto páginas web locales como remotas; esta característica permite validar el contenido web tanto en la fase de desarrollo como en las fases de mantenimiento o monitorización, cuando la página ya ha sido publicada en la web.
- Actualmente, *EvalAccess* evalúa la accesibilidad de las páginas web de acuerdo con el conjunto de pautas de WCAG, aunque las pautas a validar son fácilmente modificables, ya que no están integradas en el propio código de la herramienta.
- Dado que *EvalAccess* ha sido diseñada como un *web service,* se puede integrar fácilmente en otras aplicaciones, como por ejemplo herramientas de autor o de diseño web.

9.3.3.2 Arquitectura de la herramienta. Tal como se puede apreciar en la Figura 9.2, la arquitectura de la herramienta de validación *EvalAccess* consta de 3 niveles [58].

Fig. 9.2 Arquitectura de *EvalAccess*

NIVEL 1: Buscador de Pautas GXML. Este nivel se encarga de analizar el código de la página web a validar y de obtener las pautas aplicables desde la base de datos de las pautas de accesibilidad GDB (*Guidelines Data Base*).

En la base de datos GDB se guardan las pautas de accesibilidad formateadas de acuerdo a una gramática desarrollada específicamente con ese propósito. La gramática ha sido definida mediante un *XML Schema* [59] lo que permite especificar las pautas en un lenguaje basado en XML [29] denominado GXML (*Guidelines in XML*). En la Tabla 9.1 se pueden ver los elementos principales de una pauta definida en este lenguaje y un ejemplo de una pauta formateada.

Tabla 9.1 Ejemplo de una pauta formateada

Elemento	Descripción	Ejemplo
GuidelineID	Identificador de una pauta	*5*
Title	Título de la pauta	*Create tables that transform gracefully.*
Description	Descripción de la pauta	*Ensure that tables have necessary mark-up to be transformed by accessible browsers and other user agents.*
URL	Dirección donde se puede encontrar más información	*http://www.w3.org/TR/WCAG10/#gl-table-markup*
CheckpointID	Identificador del punto de verificación	*5*
CPDescription	Descripción del punto de verificación	*Provide summaries for tables.*
Priority	Prioridad del punto de verificación	*3*
EvaluationType	Tipo de validación que se aplica al punto de verificación: manual o automática	*Automatic*
Keyword	Elemento del código HTML a evaluar	*TABLE*
KevType	Tipo de validación a aplicar al elemento HTML	*AnalyseAttributes*
Attribute	Atributo del elemento HTML a evaluar	*SUMMARY*
AevType	Tipo de validación aplicable al atributo	*NoEmpty*

Gracias a esta característica de *EvalAccess* se pueden definir nuevas versiones de las pautas o modificar las existentes de una forma simple.

NIVEL 2: Intérprete de Pautas GXML. Este nivel se encarga de interpretar las pautas devueltas por el nivel inferior y de evaluar su cumplimiento en la página web. Los errores detectados son devueltos al nivel superior.

NIVEL 3: Creador de informes RXML. Este nivel se encarga de recoger todos los errores detectados por el nivel inferior en un informe completo que será el resultado de la validación. Este informe también está formateado en XML mediante el lenguaje RXML (*Report in XML*) definido para este fin. Los principales elementos de este lenguaje se describen en la Tabla 9.2.

Tabla 9.2 Ejemplo de informe de errores detectados

Elemento	Descripción	Ejemplo
GuidelineID	Identificador de la pauta que no se cumple	*1*
CheckpointID	Identificador del punto de verificación que no se cumple	*1*
CPTitle	Título del punto de verificación	*Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content).*
Priority	Prioridad del punto de verificación	*1*
URL	Dirección donde se puede encontrar más información	*http://www.w3.org/TR/WAI-WEBCONTENT/#gl-provide-equivalents*
Lines	Líneas en el código HTML donde se repite el mismo error	*2,18,56*
Keyword	Elemento de HTML que hay que revisar para corregir el error	*IMG*
Attribute	Atributo del elemento HTML donde se ha detectado el error	*ALT*

Por otro lado, para poder satisfacer la fácil integración de *EvalAccess* en cualquier otra aplicación (v.g. en una herramienta de autor) se ha implementado como un *web service* [60]. Por tanto, cualquier aplicación puede acceder a los servicios ofrecidos por *EvalAccess* mediante el protocolo SOAP (*Simple Object Access Protocol*). De este modo se pueden implementar diferentes aplicaciones cliente (en cualquier lenguaje de programación) que hagan uso de *EvalAccess* mediante llamadas remotas a procedimientos (RPC, *Remote Procedure Call*).

9.3.3.3 Proceso de validación. Como el *web service* sólo puede ser accedido desde una aplicación, se ha diseñado una interfaz de usuario que actúa como cliente del *web service EvalAccess* [50]. De este modo cualquier usuario puede acceder directamente a la herramienta sin necesidad de disponer de una aplicación cliente. Esto ha permitido realizar diversas pruebas y evaluaciones de la propia herramienta. De hecho, la versión actual de la interfaz de usuario ha sido rediseñada como resultado de un estudio de su usabilidad [61].

Así pues, mediante esta interfaz, un usuario puede validar la accesibilidad de una página web, bien dando como entrada el código HTML o bien la URL de la página a validar. Una vez introducido ese dato, la interfaz actúa como cliente e invoca al procedimiento correspondiente de *EvalAccess*. A partir de ese momento, los pasos que *EvalAccess* sigue para proceder con la validación de la accesibilidad son los siguientes:

1. Obtener el código HTML, si se trata de una página web remota.
2. Analizar el código basándose en los elementos HTML.
3. Buscar las pautas de accesibilidad aplicables en la base de datos GDB, para cada elemento HTML.
4. Interpretar las pautas aplicables y evaluar el elemento según las reglas descritas en ellas.

5. Generar el informe con los errores encontrados en el proceso en el formato RXML.
6. Enviar el informe al cliente.

El cliente recibe el informe formateado según RXML y lo transforma adecuadamente para mostrar los resultados de la validación al usuario.

9.3.4 Métodos de medición de la eficiencia de herramientas de validación

La medición de la **corrección** o ***correctness*** (verificación correcta de las validaciones de las pautas de accesibilidad) y **completitud** o ***completeness*** (consideración de todos los aspectos relevantes de la accesibilidad) es esencial para determinar la **eficiencia de una herramienta automática de validación**.

La corrección se puede entender como la capacidad para realizar la validación de cualquier página web minimizando el número de **falsos positivos** (aquellos aspectos que son marcados como errores aun siendo correctos en realidad) producidos. Por tanto, está relacionada con la precisión con la que efectúa la validación de las pautas de accesibilidad implementadas.

Por otro lado, la completitud está relacionada con el número de test de accesibilidad implementados en la herramienta. Si un error de accesibilidad real en una página web no es detectado como tal, se dice que la herramienta ha producido un **falso negativo**.

Actualmente, no todos los errores de accesibilidad pueden ser automáticamente detectados, ya que la validación de algunas pautas de accesibilidad requiere evaluación manual por parte del usuario. Por ejemplo, una pauta de este tipo sería "*Usar el lenguaje más claro y simple apropiado para el contenido del sitio web*", pauta que de momento no puede ser automáticamente evaluada. La verificación de este tipo de pautas de accesibilidad normalmente se introduce en el informe de resultados de la validación como un error potencial que requiere evaluación manual para determinar su cumplimiento. Por tanto, este tipo de errores no afecta al número de falsos negativos producido por la herramienta y no se tienen en cuenta a la hora de evaluar su eficiencia.

Una herramienta de verificación será correcta y completa cuando la comprobación del cumplimiento de todas las pautas de accesibilidad automáticamente verificables está implementada correctamente. Así pues, para poder medir su eficiencia será necesaria una medición exacta de los falsos negativos y falsos positivos generadas por ella.

Existen diferentes métodos para medir la eficiencia de las herramientas de validación de la accesibilidad. Uno de estos métodos, utilizado en diversos trabajos [62, 63], consiste en evaluar una serie de sitios web con diferentes herramientas y comparar los resultados obtenidos para poder apreciar las diferencias en los resultados. Otro método consiste en desarrollar un conjunto de ficheros de prueba, en este caso páginas web, que cubran todos los posibles errores de accesibilidad, evaluarlos con la herramienta de validación y analizar los resultados. Este último método ha sido el elegido por el centro de investigación *Adaptive Technology Resource Center* de la Universidad de Toronto [64], a la hora de desarrollar el banco de pruebas *Access Tool Reviewer* (ATR) [65]. Esta herramienta engloba 260 ficheros

de prueba, cada uno de ellos relacionado con un aspecto de accesibilidad concreto y permite al usuario registrar el resultado obtenido en la evaluación de estos ficheros por cualquier herramienta de validación en términos de si ha detectado algún error o no. Algunos de estos ficheros han sido diseñados para detectar un falso negativo, es decir, contienen un error real de accesibilidad y otros, sin embargo, contienen elementos accesibles con el fin de detectar posibles falsos positivos.

La ventaja del primer método es que las evaluaciones se realizan en sitios web reales, por tanto, los resultados serán relativos a problemas del "mundo real". No obstante, los sitios web seleccionados para la evaluación podrían no contener todos los posibles errores detectables automáticamente. Por lo que, aunque evalúe correctamente la corrección de la herramienta (ciñéndose a las pautas afectadas en los sitios web seleccionados), no es completamente fiable para determinar la completitud de la herramienta.

El segundo método serviría tanto para determinar la corrección de la herramienta como la completitud, siempre y cuando los ficheros de prueba abarcaran todos los posibles errores detectables automáticamente. De todos modos, desarrollar un conjunto completo de ficheros de prueba es una tarea compleja y de ella depende la calidad de la medición de la eficiencia de la herramienta de validación.

En consecuencia, se considera que para que la medición de la eficiencia de una herramienta de validación de la accesibilidad aporte un resultado fiable debe combinar ambos métodos.

9.4 Metodología de desarrollo de aplicaciones web accesibles

Para el desarrollo de aplicaciones web accesibles es esencial tanto la adopción de **políticas internas de accesibilidad** dentro de la organización, como la definición de una metodología de desarrollo adecuada. La metodología propuesta tiene como objetivo la integración de la accesibilidad en el ciclo de vida de una aplicación web, lo que implica el desarrollo de aplicaciones accesibles para el mayor número de usuarios posibles. Por otro lado, la adecuada implantación de esta metodología debe permitir una gestión de la accesibilidad metódica, fácil de mantener y de calidad.

9.4.1 La accesibilidad en las organizaciones

Para poder desarrollar aplicaciones web accesibles es fundamental la correcta implantación de políticas de accesibilidad en la organización que las desarrolla y mantiene, lo que requiere tener en cuenta los recursos disponibles en la organización así como su conocimiento de la accesibilidad. En este apartado se describen los diferentes tipos de políticas de accesibilidad que se pueden desarrollar en una organización para después definir los pasos necesarios con el fin de poder implantarla correctamente.

9.4.1.1 Políticas internas de accesibilidad. El compromiso de una organización o empresa para ofrecer servicios web que cumplan las pautas de accesibilidad requiere del desarrollo de políticas internas de accesibilidad [66]. Estas políticas internas deben tener como objetivo establecer un proceso de desarrollo que integre la accesibilidad como característica esencial del producto y la evaluación continua de dicho proceso.

En el caso de que la organización sea una empresa o asociación que no desarrolle aplicaciones web pero que se compromete a implementar y mantener su sitio web accesible la política interna de accesibilidad puede ser simple. Sin embargo, la política a implantar en una empresa que desarrolle aplicaciones web deberá ser más compleja, ya que, en este caso, habría que establecer toda una metodología de desarrollo de producto que incorpore la accesibilidad como característica esencial. Será todavía más complicada cuando el proceso de desarrollo implica a terceras partes, es decir, otras organizaciones que desarrollen parte del contenido o de las aplicaciones web.

En el primer caso bastaría con la definición de una política interna que especifique las pautas de accesibilidad que se van a tomar en cuenta (v.g. las WCAG) y el nivel de cumplimiento de estas pautas, (v.g. A, Doble-A o Triple-A). Un ejemplo de una política interna de accesibilidad simple podría ser la siguiente:

"*[Esta organización] se compromete a asegurar la accesibilidad de su sitio web. Todas las páginas del sitio web cumplirán las pautas de W3C/WAI Web Content Accessibility Guidelines 1.0, con un nivel de cumplimiento de Doble-A para la fecha [fecha]*".

Si la actividad de una empresa u organización es la del desarrollo de aplicaciones web es necesario asegurar que todas las aplicaciones desarrolladas cumplen las pautas de accesibilidad, además de los contenidos web proporcionados por terceras partes, empresas subcontratadas u organizaciones externas. El usuario tiene que acceder al contenido desarrollado por la empresa y al desarrollado por empresas subcontratadas con la misma facilidad. En este caso, un ejemplo de política podría ser:

"*[Esta organización] se compromete a asegurar la accesibilidad de las aplicaciones web desarrolladas. El contenido web producido por nuestra organización, nuevo o modificado, y el desarrollado por nuestras organizaciones colaboradoras cumplirá con el nivel de conformidad Doble-A de las pautas de W3C/WAI Web Content Accessibility Guidelines para la fecha [fecha]. Además, se iniciará un proceso de monitorización de la accesibilidad. Se requerirá a nuestros proveedores de software de desarrollo de aplicaciones web, información acerca de su cumplimiento con las pautas de W3C/WAI Authoring Tool Accessibility Guidelines para la fecha [fecha]. Nuestro sitio web incluirá un enlace a esta política interna. Revisaremos esta política en el futuro para considerar su modificación a nuevas versiones de las pautas de accesibilidad existentes*".

La implicación de terceras partes en el desarrollo de políticas internas sobre accesibilidad requiere un esfuerzo adicional para adiestrar a estos desarrolladores de contenido web en el cumplimiento de las pautas especificadas en la política interna de la organización. A veces es necesario especificar en la política interna hasta qué punto se va a implicar a las empresas subcontratadas y a otras organizaciones proveedoras de contenido. Por ejemplo:

"*Esta política interna es aplicable al contenido web producido o modificado por [la organización]. Además, esta organización está desarrollando una serie de acciones para asegurar la accesibilidad del contenido proporcionado por los desarrolladores subcontratados o colaboradores:*

- *Informar de la política interna de accesibilidad desarrollada en esta organización.*
- *Proporcionar enlaces a información sobre cómo implementar la accesibilidad web.*
- *Incentivar el desarrollo accesible.*
- *Monitorizar y proporcionar información sobre el contenido web inaccesible que han producido.*

Si los desarrolladores de contenido web de empresas subcontratadas o colaboradores no acceden a producir contenido web accesible para la fecha [fecha], la organización buscará desarrolladores alternativos".

En algunos casos, y dependiendo de la situación de la organización o empresa respecto a la accesibilidad, será necesario aplicar una política interna de accesibilidad incremental, es decir, se podría establecer una fecha para el cumplimiento del nivel A de las pautas de accesibilidad y otra posterior para el cumplimiento del nivel Doble-A y así sucesivamente. También al principio se podría aplicar esa política sólo en una serie de áreas del sitio web y luego extender su aplicabilidad a las restantes.

Se han descrito hasta ahora las posibles políticas internas que una organización puede adoptar pero como todo cambio en el proceso de desarrollo del producto, la incorporación de políticas de integración de la accesibilidad también requiere de un plan de implantación que se describirá en el siguiente apartado.

9.4.1.2 Plan de implantación de políticas de accesibilidad. Tal como se ha visto, el plan de implantación de políticas de accesibilidad [67] puede variar según el tipo de organización. A continuación se describen algunos de los pasos necesarios para su implantación, aunque el orden de los mismos puede también variar según la organización en que se aplique.

Establecer responsabilidades

Para la implantación de una nueva política interna siempre es importante establecer responsabilidades dentro de la organización. Así, sería interesante que además de identificar un responsable general sobre accesibilidad web se estableciera un grupo de coordinación para llevar a cabo el plan de implantación de políticas de accesibilidad.

Realizar una evaluación inicial

Antes de definir una determinada política interna de accesibilidad es esencial realizar una evaluación inicial de la organización. El resultado de esta evaluación inicial determina el tipo de política interna a aplicar. Así, si se manifiesta que la conciencia de la necesidad de la accesibilidad web está lo bastante extendida en la organización, se podrán adoptar políticas más restrictivas, es decir, que exijan mayor grado de cumplimiento.

Lo primero de todo será averiguar si la organización está sujeta a requisitos externos sobre accesibilidad web, por ejemplo, legislación sobre accesibilidad en

vigor. Si es así, la política interna a definir será, como mínimo, tan restrictiva como la legislación en vigor.

Para poder evaluar el grado de conocimiento de la accesibilidad en la organización, se tendrán que llevar a cabo entrevistas y evaluaciones tanto del sitio web de la organización como de la metodología de desarrollo aplicada a la implementación del contenido web. Se tendrá que evaluar especialmente el conocimiento de los desarrolladores web sobre diseño de contenido web accesible y el tipo de soporte a la accesibilidad que ofrece el *software* de desarrollo utilizado actualmente.

Desarrollo y promoción de la nueva política interna de la organización

Según los resultados obtenidos en la evaluación inicial, se establecerá una política organizacional interna sobre accesibilidad web. Esta política interna se incluirá en las políticas existentes sobre diseño web y tecnologías.

Además de notificar a toda la organización la creación de la nueva política interna, se tendrá que desarrollar un plan de promoción inicial y continua para así poder formatear el conocimiento sobre la nueva política organizacional, tanto interna como externamente a los clientes y proveedores de contenido web.

Seleccionar software apropiado

Para poder llevar a cabo la implantación de una política sobre accesibilidad es necesario seleccionar *software* de desarrollo adecuado. Para ello, habrá que seleccionar *software* que cumpla con las pautas ATAG e instalarlo usando la configuración recomendada para desarrollo de contenido accesible. En [68] se pueden encontrar recomendaciones para la selección de *software* de desarrollo de aplicaciones web. Además, el grupo de trabajo de la WAI *Authoring Tool Accessibility Guidelines Working Group* (AUWG) realiza periódicamente revisiones del *software* de desarrollo existente para evaluar su cumplimiento de las pautas ATAG [69].

Como se ha visto anteriormente, existen aplicaciones que evalúan y ayudan en la reparación de los errores de accesibilidad que serán útiles para la evaluación, corrección y **monitorización de la accesibilidad** de las aplicaciones desarrolladas. Será necesario seleccionar las herramientas de este tipo que más se ajusten a las necesidades de la organización. Se puede encontrar información sobre diferentes herramientas existentes en [70].

Por último es muy recomendable el uso de una aplicación donde se puedan publicar los errores o carencias encontradas en el *software* seleccionado o los errores habituales cometidos en el desarrollo de contenido web accesible.

Proporcionar formación

Planificar varias opciones de formación que puedan satisfacer las necesidades de las personas con diferentes responsabilidades en la organización es imprescindible para poder desarrollar adecuadamente una política de accesibilidad. Se pueden encontrar recomendaciones sobre cómo preparar la formación según el tipo de audiencia en [71, 72].

Así mismo es importante que a la hora de planificar la formación se tengan en cuenta los cambios de responsabilidades y de personal en la organización, de forma que los diferentes cursos de formación se impartan con la frecuencia necesaria.

Integrar la accesibilidad en la metodología de desarrollo
Para poder implantar correctamente una política de accesibilidad interna, se tendrá que integrar la accesibilidad en la metodología de desarrollo de contenido web. Después de evaluar las carencias detectadas en la evaluación inicial de la organización, se tendrá que diseñar e implantar una metodología cuya prioridad sea la accesibilidad. En el apartado 9.4.2.1 se describe una posible metodología de desarrollo.

Monitorizar la implantación de accesibilidad
Tan importante como la implantación de políticas internas de accesibilidad es la monitorización del proceso de desarrollo, ya que si se realizan evaluaciones o monitorizaciones regulares se podrán detectar las posibles carencias del proceso de desarrollo implantado, posibilitando así reforzar la política de accesibilidad adoptada. Por tanto, es esencial especificar y planificar evaluaciones periódicas que determinen la calidad del proceso de desarrollo, así como revisiones de todos los aspectos del plan de la implantación de la política interna. Estas revisiones también se extenderán al contenido desarrollado por empresas o desarrolladores subcontratados, comunicándoles después los resultados obtenidos.

Si se decidió adoptar una política interna de accesibilidad incremental, esta fase servirá para determinar el momento en que la organización está preparada para pasar a la siguiente fase, es decir, para adoptar una política más restrictiva en cuanto al nivel de cumplimiento requerido o las áreas de desarrollo afectadas.

También es importante en este proceso invitar a los usuarios a dar su opinión sobre el contenido web desarrollado por la organización, por ejemplo, estableciendo vías de comunicación con los usuarios para darles respuesta a sus peticiones y recoger sus opiniones o críticas.

9.4.2 Integración de la accesibilidad en el proceso de desarrollo

El objetivo de este apartado es dar a conocer los aspectos de accesibilidad a tener en cuenta en cada fase del ciclo de vida de una aplicación web: estudio de la factibilidad, análisis, diseño, implementación, evaluación y mantenimiento. Se trata de determinar y describir las decisiones que se tienen que tomar en cada fase para que el producto cumpla con las pautas que propician una web accesible y sin obstáculos.

En la fase de estudio de la factibilidad del producto, se debe hacer una evaluación preliminar para poder determinar si la organización es capaz de proceder a la consecución de sus objetivos. Esta primera evaluación debe diagnosticar las limitaciones técnicas, económicas y de recursos que la entidad tenga en el ámbito de la accesibilidad web. También se incluye aquí un análisis que determine la capacidad que la entidad tiene para hacer frente a dichas carencias y que señale de una forma detallada en qué modo se van a solventar. Uno de los principales obstáculos que se

suele encontrar en esta fase es la falta de una política interna que tenga como propósito el desarrollo de una aplicación web accesible. La escasez de personal con una formación adecuada y la carencia de responsables en el tema son los mayores inconvenientes a los que puede enfrentarse cualquier organización. Todos estos impedimentos se resuelven llevando a cabo políticas que den pie a paliar esta problemática inicial, lo cual supone un gasto de recursos que la entidad tiene que sopesar en esta primera fase. Por otra parte, es indispensable un análisis de las herramientas disponibles para evaluar, desarrollar, corregir y monitorizar automáticamente la accesibilidad. Este análisis es crucial puesto que se decide qué aplicaciones se van a usar en las diferentes fases del ciclo de vida. Por tanto, un examen detallado puede ahorrar futuros contratiempos. Lamentablemente, la falta de recursos económicos para adquirir las herramientas y de profesionales que sepan usarlas adecuadamente puede frenar la consecución de la meta propuesta en un principio.

También es crucial adoptar la filosofía de "diseño universal" en la fase de análisis para evitar que se generen perfiles de usuario que excluyan a otros usuarios potenciales, lo que puede propiciar que la página sea inaccesible. Estas pautas afectan positivamente en la especificación de las características de los usuarios y del entorno de operación de éstos puesto que evitan la creación de limitaciones innecesarias. Es por ello por lo que se aboga por la adopción de directrices aceptadas por un gran número de usuarios y desarrolladores como son las pautas WCAG 1.0 o la *Section 508* [15]. Gracias al uso de este tipo de pautas, el análisis de las tareas que los usuarios vayan a ejecutar produce unas especificaciones y requisitos simples, intuitivos, flexibles e igualitarios.

La salida de la fase de análisis se formaliza en el diseño. De acuerdo a las pautas adoptadas existen técnicas asociadas tales como las (TWCAG 1.0), que posibilitan la conformidad con las pautas. Estas técnicas pretenden ser independientes de la tecnología que se vaya a usar a la hora de desarrollar una aplicación web. Por tanto, hay que identificar las técnicas necesarias en cada caso para que las especificaciones recabadas en la fase de análisis cumplan los principios de diseño accesible.

Las restricciones técnicas asumidas en etapas anteriores deben tenerse en cuenta en la fase del diseño. Los impedimentos mas reseñables que afectan a la accesibilidad son la escasez de herramientas de autor con soporte para diseño accesible. Este déficit es uno de los mayores obstáculos para generar contenido accesible.

Una vez superada la fase de diseño, la producción o implementación del producto se convierte en la principal actividad. Los principios de diseño universal mencionados anteriormente deben de estar presentes continuamente en la mente de los desarrolladores. Las herramientas de autor que cumplan con las pautas de accesibilidad ATAG ayudan a plasmar estos principios en el producto.

A la hora de generar contenido en esta primera fase, y más concretamente al generar contenido multimedia, es necesario proveer de formatos equivalentes cuando proceda. Su finalidad es satisfacer las pautas de accesibilidad web en uno de sus puntos más relevantes: la producción de contenido equivalente que proporcione al usuario la misma funcionalidad que el original. Sirva como ejemplo la adición de un archivo equivalente en texto cuando en el diseño original se haya incluido un archivo de audio.

Dentro del ciclo de vida de aplicaciones web que integran el desarrollo de contenido accesible, la fase de evaluación tiene una gran envergadura. Es en esta fase donde se comprueba si el análisis, diseño e implementación orientada al diseño universal ha cumplido con éxito las especificaciones. Para alcanzar esta resolución hay que emplear herramientas de validación, reparación y transformación automática de la manera en la que se especifica en el apartado 9.2.4.1 **Metodología de validación de la accesibilidad**. Es de gran ayuda contar con una plataforma que permita publicar los errores cometidos para no volver a caer en ellos de nuevo y así depurar la técnica de desarrollo.

Además, todo proceso de desarrollo debe tener como objetivo la obtención de productos de calidad. Es en la fase de evaluación donde, además de verificar el cumplimiento de las pautas de accesibilidad, será también necesaria la verificación de la calidad del producto desarrollado. Esta propiedad del producto está compuesta por diversos atributos donde la accesibilidad también juega un papel importante. Este aspecto es tratado en el apartado 9.4.4.

Uno de los aspectos críticos en el ciclo de vida de aplicaciones accesibles es el mantenimiento. Debido al carácter eminentemente dinámico de la Web, los contenidos de las aplicaciones web varían con relativa frecuencia. Desde el punto de vista de la accesibilidad, se entiende la fase de mantenimiento como una etapa de monitorización donde se va a controlar la evolución de la accesibilidad. Por mucho que se cumplan las especificaciones, es difícil cuantificar la progresión o regresión de la accesibilidad porque normalmente se refieren a **métricas cualitativas** como los niveles de conformidad. Es por ello conveniente contar con **métricas cuantitativas** que midan con más exactitud la evolución de la accesibilidad. Sea una evolución positiva o negativa, se han de publicar los factores y determinaciones que han llevado a la aplicación a ese estado. Si ha habido una involución en accesibilidad habrá que acotar los fallos para no reincidir en los mismos errores. Una herramienta que mida automáticamente la accesibilidad en términos numéricos con una frecuencia predeterminada y que a su vez publique informes, es una aplicación potente, valiosa e indispensable en el contexto de esta fase del ciclo de vida.

La adopción de las decisiones relativas a la accesibilidad en el ciclo de vida de una aplicación web, tiene como objetivo principal el desarrollo de aplicaciones para el mayor número de usuarios posibles. Por otro lado, permite una gestión de la accesibilidad metódica, fácil de mantener y de calidad.

9.4.2.1 Metodología de validación de la accesibilidad. En el proceso de desarrollo y mantenimiento de aplicaciones web accesibles son de vital importancia las evaluaciones periódicas de accesibilidad que deben llevarse a cabo siguiendo una metodología rigurosa si se quiere que sean efectivas. La metodología de validación tiene que ser completa, es decir, debe evaluar todos los aspectos de accesibilidad y, además, determinar los pasos específicos a seguir.

En este apartado se propone una metodología para la **evaluación de la accesibilidad** de aplicaciones web basada en las recomendaciones descritas en [73]. Esta metodología combina la evaluación de accesibilidad automática, manual y con usuarios. Estos tres diferentes tipos de evaluaciones de accesibilidad son necesarios para poder identificar la mayoría de errores de accesibilidad y por tanto, es imprescindible incluirlos para que la metodología sea completa. También es cierto

que la combinación de los diferentes tipos de evaluación requiere la adquisición de diversos conocimientos por parte del evaluador, ya que será necesaria la familiaridad con los lenguajes de etiquetado, las pautas de accesibilidad existentes y técnicas para corregir errores, las herramientas automáticas de validación de accesibilidad, las opciones de configuración de los navegadores web, la experiencia en el uso de tecnologías asistentes y la coordinación con grupos de usuarios con diferentes necesidades. Por tanto, para poder llevar a cabo las evaluaciones de accesibilidad del contenido producido por una organización es aconsejable establecer un grupo de evaluadores que conjuntamente tengan los conocimientos suficientes, lo cual mejorará la calidad de las evaluaciones. Se puede encontrar más información sobre el modo de funcionamiento de los grupos de evaluación en [74].

A continuación se describen los pasos a seguir para llevar a cabo una evaluación completa de la accesibilidad de los sitios web:

Identificar el nivel de conformidad que debe cumplir el sitio web.
El nivel de conformidad que debe cumplir el sitio web está determinado por la política interna de accesibilidad aplicada en la organización y por las especificaciones del producto a desarrollar.

Seleccionar e identificar las páginas web a evaluar.
Si el sitio web consta de un gran número de páginas web habrá que seleccionar una muestra adecuada para poder evaluarlas de forma manual y con usuarios. Esta selección de páginas web tendrá que ser representativa, es decir, englobará por lo menos una página de cada tipo y las que supongan mayor interacción con el usuario o aquellas que se espera que vayan a ser más visitadas.

A la hora de realizar la evaluación automática se tendrán en cuenta las direcciones de todas las páginas del sitio web. Si el sitio web consta de demasiadas páginas se definirá una muestra suficiente y representativa de páginas. Esta muestra contendrá aquellas páginas pertenecientes a diferentes secciones, diferente diseño, páginas generadas dinámicamente, producidas usando diferentes pautas, páginas críticas para la organización, etc.

Realizar la evaluación automática del sitio web.
En este paso se utilizarán diferentes herramientas automáticas de validación tales como las descritas en la sección 9.3. Estas herramientas se utilizarán tanto para validar la corrección de la sintaxis del lenguaje de etiquetado y de las hojas de estilo como para evaluar la accesibilidad de las páginas web. Se recomienda validar las páginas seleccionadas en el paso anterior al menos con dos herramientas de validación diferentes (ya que el resultado puede variar según la herramienta) y todas las páginas web del sitio con al menos una herramienta de validación.

Realizar la evaluación manual del sitio web.
Después de realizar la evaluación automática del sitio web habrá que evaluar manualmente aquellos puntos de verificación que no puedan ser evaluados automáticamente y sean relevantes para el sitio web. Como mínimo aquellos puntos

de verificación que se han de cumplir para el nivel de conformidad identificado en el primer paso.

También es necesario examinar la selección de páginas con diferentes navegadores web, diferentes versiones del mismo navegador y en diferentes plataformas. Para todas estas configuraciones habrá que probar lo siguiente:

- Analizar si existen equivalentes de texto apropiados para las imágenes, al acceder sin cargar imágenes de la página web.
- Asegurar que el contenido audio también está disponible como equivalentes de texto, al acceder sin cargar archivos de sonido.
- Usar las opciones de navegador para cambiar el tamaño de la fuente y verificar que el texto de la página cambia correctamente de tamaño y que la página todavía es usable con una fuente de tamaño mayor.
- Probar con diferentes resoluciones de pantalla y cambiando de tamaño la ventana del navegador para verificar que no es necesario utilizar el *scroll* horizontal. Ello sirve para verificar que en el código fuente de las páginas no se especifican tamaños o posiciones absolutas.
- Visualizar las páginas en escala de grises, desactivando colores o de lo contrario imprimir las páginas en escala de grises. Así se podrá verificar que el contraste entre colores es adecuado.
- Navegar por las páginas web, especialmente las que contengan formularios, usando sólo el tabulador, es decir, sin usar el ratón para así verificar que se puede llegar a cualquier enlace y control de formularios. Asegurar también que se indica claramente cuál es el destino de cada enlace.
- Examinar las páginas tras desactivar *scripts*, hojas de estilo y *applets* y verificar que se pueden usar todas las funcionalidades mediante mecanismos alternativos.

Además será necesario el uso de navegadores de texto (v.g. el navegador de texto *Lynx* [75]), lectores de pantalla (v.g. la herramienta *Jaws* [76]) y navegadores por voz (v.g. *IBM Home Page Reader* [77]) para verificar que todas las funciones y toda la información equivalente está disponible también con un navegador de texto o por voz, tal como lo está con un navegador gráfico. También habrá que verificar que la información está presentada en el orden adecuado al leer en un navegador de texto o reproducir en un navegador por voz.

Para terminar con la evaluación manual del sitio web, habrá que leer el contenido para determinar si el texto es claro y simple. También se puede obtener la evaluación de este aspecto de accesibilidad en el siguiente paso de la evaluación.

Realizar una evaluación de la accesibilidad del sitio web con usuarios.

Para este paso hace falta coordinar a usuarios con diferentes discapacidades, niveles de experiencia técnica, niveles de familiaridad con el sitio web a evaluar y que usen diferentes tecnologías asistenciales. Estos usuarios deberán evaluar las páginas web seleccionadas y navegar por todo el sitio web para así poder identificar los problemas reales de dicho sitio. A veces, esta evaluación se efectúa en entornos controlados, como por ejemplo en laboratorios específicos, pero también es valioso que los usuarios evalúen el sitio web en su entorno normal de navegación. Es recomendable

utilizar alguna de las técnicas de evaluación consolidadas para el estudio de usabilidad de sitios web [78, 79]

Documentar los resultados
Al final de todo este proceso es necesario documentar todos los errores encontrados, el método con el que se han encontrado y las páginas que han sido evaluadas. Si se ha excluido de la evaluación algún área del sitio web también es importante especificarlo en el documento de resultados de la evaluación, ya que cuanto más detallado sea éste, más fácil será la futura corrección de los problemas encontrados. Es interesante que la organización tenga una plantilla para la elaboración de la documentación de este proceso que además servirá para guiar a los evaluadores en los pasos que hay que seguir en la evaluación. En [80] se ofrecen recomendaciones para elaborar una plantilla de resultados.

Evaluación de páginas web generadas dinámicamente
Normalmente las páginas generadas dinámicamente suelen tener la misma presentación definida en una plantilla que se completa con el contenido obtenido automáticamente desde una base de datos. En este caso no es suficiente con evaluar la accesibilidad de las plantillas, ya que el contenido generado automáticamente podría contener código que también necesita ser evaluado. Por tanto, además de seleccionar varias páginas generadas dinámicamente, es decir, la combinación de plantillas con contenido, para formar parte de la muestra representativa de páginas web del sitio también deben evaluarse las plantillas y el contenido de la base de datos por separado.

La accesibilidad de las plantillas se evaluará como el resto de páginas web y será necesario evaluar el contenido de la base de datos para verificar si el diseño de la misma permite que toda la información necesaria para generar páginas accesibles (v.g. equivalentes de texto para imágenes o audio) esté registrada.

9.4.2.2 Monitorización de la accesibilidad. Tal como se ha mencionado anteriormente, la fase de mantenimiento se entiende en este contexto como la monitorización de la evolución de la accesibilidad. Este proceso comienza una vez que la fase de evaluación da por cumplidos los prerrequisitos y especificaciones de un producto y se publica la aplicación en la web. En esta fase, hay que llevar a cabo evaluaciones de accesibilidad con una periodicidad determinada. Además, se debe evaluar la accesibilidad tanto si se ha hecho alguna modificación de la aplicación como si se ha insertado alguna nueva sección, ya que estos cambios podrían afectar al grado de satisfacción de las pautas de accesibilidad. El resultado de las evaluaciones de accesibilidad ejecutadas en esta fase determinará la evolución de la accesibilidad del producto durante su ciclo de vida.

Una correcta medición de la accesibilidad pondrá de manifiesto si los cambios introducidos han modificado la accesibilidad para mejor o para peor. En ambos casos, se debe publicar en el ámbito de la organización qué factores han determinado que la accesibilidad haya cambiado para tomar las medidas oportunas. Ya que es habitual que los diseñadores deban desarrollar aplicaciones web nuevas en cortos periodos de tiempo y las modificaciones se sucedan con relativa frecuencia, es interesante automatizar este proceso de monitorización, de forma que se tenga una correcta visión

de la evolución de la accesibilidad de una determinada aplicación web en cualquier instante de su ciclo de vida.

Para una monitorización eficiente y precisa de la accesibilidad, es necesario el diseño de una métrica de accesibilidad que ofrezca un valor exacto del nivel de accesibilidad de una aplicación en un instante concreto. Por tanto, una medición en términos cuantitativos es imprescindible para un cálculo más exacto y puede determinar el nivel de accesibilidad de una misma aplicación web en sus diferentes versiones de una manera más rigurosa. En el apartado 9.4.5 se tratan aspectos relacionados con métricas cuantitativas de la accesibilidad.

9.4.3 Modelo de proceso para el desarrollo web accesible

Uno de los factores determinantes a la hora de seleccionar un modelo de proceso es el tiempo disponible para desarrollar el producto. En el caso de las aplicaciones web, los desarrolladores suelen estar obligados a implementar estas aplicaciones en periodos muy breves de tiempo. Esta característica hace que sea necesaria una metodología específica para el desarrollo de aplicaciones web accesibles que establezca claramente los pasos a seguir en cada fase del ciclo de vida.

Aunque el modelo de proceso aplicable también dependerá del tamaño de aplicación a desarrollar y las características del equipo de desarrollo, de acuerdo con [81], los modelos de proceso iterativos son los que mejor se adaptan al desarrollo de aplicaciones web accesibles. En esta línea se han desarrollado algunos trabajos como [82] y [83].

Los modelos de proceso iterativos, al contrario del tradicional modelo de proceso en cascada, hacen posible el desarrollo de versiones o prototipos de la aplicación en etapas tempranas de la producción. Esta característica facilita la realización de evaluaciones de la accesibilidad del producto durante su proceso de desarrollo. Así, los errores de accesibilidad encontrados en estas evaluaciones son más fáciles y menos costosos de corregir, se previene la propagación de errores y se evitan fallos similares en el resto del proceso. Es de gran ayuda contar con una plataforma para la publicación de los errores cometidos en un ámbito interno. Esto contribuye a depurar la técnica de desarrollo porque se evita caer de nuevo en errores cometidos anteriormente.

Para poder determinar los modelo de proceso aplicables, hay que tener en cuenta que una organización se puede encontrar ante dos posibles escenarios: el desarrollo desde cero de una aplicación web o bien la mejora de la accesibilidad de una aplicación existente. El modelo de proceso que muestra la Figura 9.3 se puede apreciar cómo estas situaciones dispares toman caminos opuestos para luego converger. Las fases propuestas en el ciclo de vida enfocado a la creación de aplicaciones accesibles, se secuencian de la siguiente manera el modelo de proceso propuesto:

Fig. 9.3 Modelo de proceso del desarrollo de aplicaciones web accesibles.

Como se puede apreciar, en ambos escenarios es importante prever y planificar evaluaciones periódicas de la accesibilidad durante su proceso de desarrollo, usando la metodología de evaluación anteriormente descrita. Si el resultado de la evaluación no cumple con el objetivo marcado en las especificaciones, éste será analizado y se rediseñará la aplicación con el fin de corregir los errores. Gracias a este método se obtiene una versión mejorada del producto o prototipo en cada iteración. Cuando todas las especificaciones y pautas de accesibilidad aplicadas (según la política interna de accesibilidad adoptada en la organización o especificaciones del cliente sobre la accesibilidad que el producto deberá cumplir) hayan sido incorporadas terminará el proceso de desarrollo comenzando así la fase de mantenimiento o monitorización.

Por otro lado, con el fin de mejorar la accesibilidad de una aplicación web existente inicialmente, en proyectos de desarrollo se tendrá que realizar una evaluación inicial. Así se podrán detectar todos los problemas de accesibilidad para luego poder analizar y diseñar el modo de corregirlos. Por último, se llevará a cabo el desarrollo de las soluciones a estos problemas de una forma iterativa para poder evaluar en cada momento del desarrollo los errores de accesibilidad que persisten. De esta forma se podrán incorporar todos los mecanismos para garantizar la accesibilidad de la aplicación web y después iniciar la fase de mantenimiento.

Es en la fase de mantenimiento donde habrá que llevar a cabo evaluaciones periódicas de la accesibilidad para poder determinar si los cambios introducidos en la aplicación influyen negativamente en el nivel de accesibilidad requerido. En el caso de que los cambios efectuados disminuyan la accesibilidad del producto, se tendrá que analizar el resultado de la evaluación para así poder diseñar e implementar las soluciones de los errores detectados.

Como se puede apreciar, en un modelo de proceso para el desarrollo web accesible es de vital importancia realizar evaluaciones de la accesibilidad durante todo el ciclo de vida.

9.4.4 Accesibilidad como medida de calidad

La **`calidad de las aplicaciones web`** debe tenerse en cuenta de una manera similar al resto de las aplicaciones *software*. La norma ISO 9126 define seis cualidades que debe tener cualquier producto *software* para que sea de calidad: funcionalidad, fiabilidad, eficiencia, usabilidad, facilidad de mantenimiento y portabilidad [84]. Bajo este punto de vista, se entiende la calidad como una propiedad compuesta por una serie de atributos interdependientes [85]. El estándar ISO 9126 también define el procedimiento de evaluación de la calidad que consta de tres pasos: medición, clasificación y valoración. Es en la fase de medición donde se aplican métricas al producto para posteriormente categorizarlo en la fase de clasificación según los resultados obtenidos. Finalmente, en la fase de valoración se evalúan los resultados producidos.

La usabilidad es uno de los atributos que hay que medir, clasificar y valorar. Según la norma ISO 9126, la usabilidad se refiere a la capacidad de un *software* de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. También se hace hincapié en la facilidad que el usuario tiene para

llevar al cabo las tareas dentro de una aplicación. La accesibilidad y la usabilidad están estrechamente relacionadas, pues ambas disciplinas comparten objetivos comunes como la satisfacción, efectividad y eficiencia del *software* [86]. Según Thatcher et al. [87], se puede entender la accesibilidad como un atributo o característica inherente a la usabilidad. Aunque se han desarrollado diversos métodos y herramientas para evaluar la usabilidad, la evaluación de la accesibilidad no está lo suficientemente desarrollada porque se carecen de métodos para una medición y valoración detallada. Es por ello necesario el desarrollo de estos métodos para hacer frente a esta carencia.

Por otro lado, según las pautas WCAG 2.0, para que una aplicación web sea accesible se han de cumplir los principios de perceptibilidad, manejabilidad, comprensibilidad y robustez. Estos principios se pueden entender como los atributos interdependientes descritos en la norma ISO 9126. Por lo que se pueden definir métricas que midan el cumplimiento de cada atributo. Aunque las pautas WCAG 2.0 se refieren a la accesibilidad global de las aplicaciones web, esta clasificación basada en atributos de la accesibilidad permite una evaluación más precisa.

Sin embargo, el nivel de conformidad no es tan preciso como se desea: una aplicación que cumple todas las pautas de prioridad 1 y otra aplicación que además de eso cumple la mayoría de las pautas de prioridad 2 tienen la calificación A. Esto demuestra la poca precisión de este método de medición cualitativo. Estos criterios se basan en la suposición de que si una aplicación web no cumple una pauta de un determinado nivel es tan poco accesible como una aplicación que no cumpla todas las pautas de ese mismo nivel. El hecho de que exista aunque sea una sola barrera de accesibilidad, hace inaccesible toda la página para algunos usuarios. Sin embargo, es fundamental una medición más precisa que una validación que sólo lo haga en términos de rechazo y aceptación. De esta forma será posible reflejar la accesibilidad del producto como un atributo de la calidad del mismo.

9.4.5 Métricas cuantitativas para la accesibilidad

Tanto en la fase de monitorización para poder evaluar la evolución de la accesibilidad de una aplicación web, como en la inclusión de la accesibilidad como un atributo de la calidad del producto, se ha evidenciado la necesidad de definir métricas cuantitativas para la accesibilidad. En este apartado se proponen los aspectos principales que se deben tener en cuenta a la hora de diseñar estas métricas.

La medición cuantitativa de la accesibilidad, al igual que la medición cualitativa, se debe hacer en base a pautas relativas a la creación de contenido accesible. Este apartado se enfoca en el ámbito de las pautas WCAG. Por tanto, el objeto de medición son las pautas de los cuatro atributos que se establecen: Perceptibilidad, Manejabilidad, Robustez y Comprensibilidad (PMRC). Los parámetros a tener en cuenta a la hora de medir la accesibilidad son:

- profundidad de la página web donde se ha cometido el error dentro del sitio,
- número de veces que no se cumple una pauta,
- número de veces que se evalúa cada pauta,
- prioridad de cada error, y

· grupo al que pertenece cada error (PMRC).

A la hora de diseñar una métrica cuantitativa de la accesibilidad y relacionados con estos parámetros existen otra serie de aspectos a tener en cuenta, algunos de los cuales se describen a continuación.

Además del número de errores (valor absoluto), la métrica debe tener en cuenta el número de veces que se ha examinado cada pauta que haya dado lugar al error (valor relativo). Esto significa que aparte de los errores reales, hay que medir el número de errores potenciales. Suponiendo que la página web tenga un único elemento no accesible, una imagen sin etiquetar de acuerdo con las pautas WCAG, si la imagen fuese el único elemento de la aplicación web, la aplicación sería totalmente inaccesible pues esa imagen no cumpliría con el objetivo previsto. Sin embargo, si se encontrara en un contexto donde hubiera más imágenes y éstas sí fueran accesibles, esta página sería más accesible que la que tiene una sola imagen. Esto es debido a que el usuario tiene mayor capacidad de interactuar con la aplicación, ya que el resto de las imágenes accesibles cumplirían su función. Por tanto, la proporción de errores detectados sobre errores posibles es una medida a tener en cuenta.

La prioridad de cada error cometido se tiene que reflejar en el resultado final puesto que la incidencia en la accesibilidad de las tres prioridades afecta de manera diferente.

También debe tenerse en cuenta el tipo de pauta (estructural, contenido). Son **pautas estructurales** aquellas que se refieren a elementos que se usan para maquetar una aplicación (aunque no sea su objetivo, caso de las tablas) como los *frames*, las tablas y los objetos empotrados (v.g. *applets*, tecnología *flash*). Son **pautas de contenido** aquellas que tienen algún significado y por tanto contienen algún tipo de información como fotos, texto y sonido. Los errores asociados a pautas referidas a elementos estructurales deben tener mayor incidencia en el resultado final. Esto es debido a que, si un elemento estructural no es accesible, sus elementos de contenido dejan de ser accesibles aunque cumplan las pautas de diseño accesible (v.g., el uso de una tabla para estructurar una página dificulta la navegación mediante tecnologías de accesibilidad). Si estas tablas tienen fotos en sus celdas y éstas están etiquetadas de acuerdo con las pautas WCAG son inaccesibles de todas maneras por encontrase en un contexto inaccesible.

Si el objetivo es la medición exacta de la accesibilidad de un sitio web, hay que tener en cuenta el contexto navegacional donde se produce el error. Este factor es más ilustrativo para medir la accesibilidad de algunas aplicaciones. El impacto de una página en un nivel profundo del sitio es menor que el impacto de otra de un nivel superior porque la probabilidad de que un usuario navegue por páginas web con mayor nivel de profundidad dentro de la estructura del sitio es menor [88]. Por tanto, a mayor profundidad que se produzca el error, menor es el peso que hay que otorgarle.

La combinación acertada de todas estas medidas puede dar un valor aproximado de la accesibilidad. El resultado final debería de darse dentro de un rango de valores y precisión predeterminados. Es por ello que debe usarse una métrica que tenga en cuenta los atributos mencionados y la limitación de rango de forma que dé lugar a medidas de accesibilidad fiables y válidas.

Por otra parte, sería conveniente disponer herramientas para poder automatizar el proceso de cálculo de la accesibilidad porque la casuística de errores es muy amplia.

Es aconsejable además que la adquisición de los datos se haga también de una manera automática con la ayuda de herramientas de validación de la accesibilidad. El mayor problema para poder integrar los dos procesos (el proceso de cálculo de la métrica cuantitativa y la detección de los errores de accesibilidad) en una sola herramienta automática es que la mayoría de herramientas de validación devuelven un informe en lenguaje natural sobre los errores encontrados. Un posible informe sobre un error encontrado podría ser el siguiente:

"*La pauta 1.1 de WCAG 1.0 se ha analizado 15 veces y no se ha cumplido en 10 ocasiones. La pauta es de prioridad 1 y ha tenido lugar en la página principal*".

Para poder automatizar todo el proceso, el formato de exportación de informes de la herramienta usada debe ser entendible por la máquina para así facilitar la integración de las diferentes herramientas. En este aspecto el consorcio web, concretamente el grupo de trabajo *Evaluation and Repair Tools Working Group* de la WAI [89], está desarrollando el *Evaluation and Report Language* (EARL) [90]. Este lenguaje basado en XML tiene como objetivo la estandarización de los resultados de las herramientas de validación de la accesibilidad, para así facilitar la posible interacción entre herramientas.

9.5 Lecciones aprendidas y conclusiones

La experiencia de diseño de herramientas para la evaluación y corrección de la accesibilidad y su integración en otras herramientas de autor [91] ha permitido detectar una serie de carencias metodológicas que se ha tratado de presentar y resolver en este capítulo.

La mayoría de los problemas de accesibilidad que encuentran los usuarios con restricciones se deben al excesivamente frecuente uso de tecnologías no estándar en el diseño web. Si los sitios web utilizaran código estándar el número de problemas de accesibilidad sería mucho menor y su solución mucho más fácil [1].

Por un lado, está claro que incluso los diseñadores más motivados para enfrentarse a la actualización o creación de páginas web accesibles necesitan disponer de las herramientas adecuadas, ya que la aplicación directa de las pautas y puntos de verificación tales como los ofrecidos por la WAI resulta prácticamente imposible en cuanto el sitio web adquiere cierta complejidad. Esas herramientas deben estar integradas en el *software* de desarrollo normalmente usadas por los diseñadores. En la actualidad existen herramientas pensadas para ser usadas *a posteriori*, lo que en muchos casos resulta inútil, ya que algunos fallos de accesibilidad pueden estar ligados a la propia estructura inicial de la página y sólo podían haber sido evitados con un diseño inicial correcto. No hay que olvidar que el objetivo final ha de ser que las aplicaciones web desarrolladas sean accesibles, y no que satisfagan los requisitos de una herramienta concreta.

Adicionalmente, se ha observado la necesidad de una política general de accesibilidad dentro de la organización que desarrolla y mantiene la aplicación web. La accesibilidad es un proceso continuo que no se resuelve encargando a un técnico la revisión y actualización del sitio web.

El proceso de desarrollo de tecnología web accesible no aporta una metodología en sí mismo. Adquiere sentido cuando está incluido en otras metodologías de diseño de tecnología web, tales como las que aparecen en diversos capítulos de este libro. Una metodología seria que incorpore el diseño accesible no puede restringirlo a la decisión de aplicar de una serie de pautas de accesibilidad. Es necesario incluirlo en todas las fases del ciclo de vida desde la concepción del sistema hasta su mantenimiento.

9.6 Referencias

[1] J. Zeldman (2003). Designing with web standards. New Riders Press.
[2] S. Pemberton (2003). The kiss of the spiderbot. Interactions. Vol. 10, No. 1, pp. 44.
[3] How people with disabilities use the web: http://www.w3.org/WAI/EO/Drafts/ PWD-Use-Web/
[4] J. Abascal (2002). Human-Computer Interaction in Assistive Technology: From "Patchwork" to "Universal Design". 2nd IEEE International Conference on Systems, Man and Cybernetics. Hammamet (Tunisia), 2002.
[5] J. Abascal, A. Civit (2001). Bridging the Gap between Design for All and Assistive Devices. In C. Stephanidis (ed.): Universal Access in HCI. Towards an Information Society for All. Lawrence Erlbaum Associates, 2001.
[6] A. M. Cook, S. M. Hussey (2002). Assistive Technology: Principles and Practice (2nd Ed). Harcourt Publishing.
[7] L. Gardeazabal (2001). Aplicaciones de la Tecnología de Computadores a la mejora de la velocidad de comunicación en sistemas de Comunicación Aumentativa y Alternativa. Servicio Editorial de la UPV-EHU. Leioa, 2001.
[8] N. Garay (2001). Sistemas de predicción lingüística. Aplicación a idiomas con alto y bajo grado de flexión, en el ámbito de la Comunicación Alternativa y Aumentativa. Servicio Editorial de la UPV-EHU. Leioa, 2001.
[9] J. Abascal , N. Garay (2003). Teaching design for all in HCI. In Stephanidis C. (ed.) Universal Access in HCI. Inclusive Design in the Information Society. Vol. 4. Lawrence Erlbaum Associates, London, pp. 3-7.
[10] C. Nicolle, J. Abascal (Eds.) (2001). Inclusive Design Guidelines for HCI. Taylor & Francis.
[11] C. Farenc, J. Vanderdonckt (Eds.) (2000). Tools for Working with Guidelines. Springer.
[12] USERfit: http://www.stakes.fi/include/1-4.htm
[13] USERfit Tool: http://www.sc.ehu.es/acwusfit/
[14] Americans with disabilities Act: http://www.usdoj.gov/crt/ada/adahom1.htm
[15] Section 508 of the Rehabilitation Act: http://www.section508.gov
[16] Iniciativa "eEurope - una Sociedad de la Información para todos": http://www.csi.map.es/csi/pg8008.htm
[17] eEurope An Information Society For All. Action Plan: http://europa.eu.int/ information_society/eeurope/2002/action_plan/pdf/actionplan_en.pdf
[18] World Wide Web Consortium: http://www.w3.org/
[19] Web Accessibility Initiative: http://www.w3.org/WAI/
[20] Web Content Accessibility Guidelines: http://www.w3.org/2004/09/wai-nav/intro/wcag.html
[21] User Agent Accessibility Guidelines: http://www.w3.org/2004/09/wai-nav/intro/uaag.html
[22] Authoring Tool Accessibility Guidelines: http://www.w3.org/2004/09/wai-nav/intro/atag.html

[23] Fact Sheet for "Web Content Accessibility Guidelines 1.0": http://www.w3.org/1999/05/WCAG-REC-fact.html
[24] Auxiliary Benefits of Accessible Web Design: http://www.w3.org/WAI/bcase/benefits.html
[25] Acctiva. Análisis sobre la accesibilidad de los portales de las universidades públicas españolas: http://www.acctiva.com/recursos/AcctivaUniversidades.pdf
[26] Accessibility and Macromedia Flash MX 2004: http://www.macromedia.com/macromedia/accessibility/features/flash/
[27] Creating Accessible Content with Macromedia Dreamweaver MX 2004: http://www.macromedia.com/macromedia/accessibility/mx/dw/
[28] HyperText Markup Language (HTML) Home Page: http://www.w3c.org/MarkUp/
[29] Extensible Markup Language: http://www.w3.org/XML/
[30] XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition): http://www.w3.org/TR/xhtml1/
[31] Modularization of XHTML™: http://www.w3.org/TR/xhtml-modularization/
[32] XHTML™ 1.1 - Module-based XHTML: http://www.w3.org/TR/xhtml11/
[33] XHTML™ 2.0: http://www.w3.org/TR/xhtml2/
[34] *W3C HTML Validator Service*: http://validator.w3.org/
[35] Cascading Style Sheets: http://www.w3.org/Style/CSS/
[36] Cascading Style Sheets, level 1: http://www.w3.org/TR/REC-CSS1
[37] Cascading Style Sheets, CSS 2.1 Specification: http://www.w3.org/TR/CSS21/
[38] Cascading Style Sheets. Current work: http://www.w3.org/Style/CSS/current-work
[39] Accessibility Features of CSS: http://www.w3.org/TR/CSS-access
[40] Document Object Model: http://www.w3.org/DOM/
[41] DOM1: http://www.w3c.org/TR/1998/REC-DOM-Level-1-19981001/
[42] Document Object Model. Level 2. Core Specification: http://www.w3c.org/TR/2000/REC-DOM-Level-2-Core-20001113/
[43] Document Object Model (DOM) Level 3 Core Specification: http://www.w3c.org/TR/2004/REC-DOM-Level-3-Core-20040407/
[44] Ecma International: http://www.ecma-international.org/
[45] Ecma Standards: http://www.ecma-international.org/publications/standards/Stnindex. htm
[46] Evaluation Tools Demonstration: http://www.w3.org/WAI/EO/2004/02/tools.html
[47] Bobby™ Accessibility Tool: http://bobby.watchfire.com/
[48] WAVE 3.0 Accessibility Tool: http://wave.webaim.org/index.jsp
[49] Test de Accesibilidad a la Web: http://www.tawdis.net/
[50] EvalAccess: http://sipt07.si.ehu.es/evalaccess/)
[51] W3C CSS Validator: http://jigsaw.w3.org/css-validator
[52] WDG HTML Validator: http://www.htmlhelp.com/tools/validator/
[53] WAI Resources: http://www.w3.org/WAI/Resources/
[54] Tidy: http://tidy.sourceforge.net/
[55] AccRepair: http://www.hisoftware.com/access/repair.html
[56] A-Prompt: http://aprompt.snow.utoronto.ca/
[57] A. Bereikdar, J. Vanderdonckt, M. Noirhomme-Fraiture (2003). KWARESMI – Knowledge-based Web Automated Evaluation Tool with Reconfigurable Guidelines Optimization. HCI International 2003. Lawrence Erlbaum Associates. Pp. 1504-1508.
[58] J. Abascal, M. Arrue, I. Fajardo, N. Garay, J. Tomás (2003). Use of Guidelines to automatically verify web accessibility. International Journal on Universal Access in the Information Society (UAIS). Special Issue on "Guidelines, standards, methods and processes for software accessibility". Vol. 3, No. 1, 71-79.
[59] XML Schema: http://www.w3.org/XML/Schema
[60] B. Galbraith, A. Tost, R. Irani, J. Basha, M. Hendricks, T. Modi, J. Milbery, S. Cable (2002). Professional Java Web Services. Wrox Press Ltd.

[61] J. Abascal, M. Arrue, D. del Río, I. Fajardo, N. Garay (2004). Estudio de Usabilidad basado en Expertos: Revisión de Guías de Diseño y Evaluación Heurística de un Servicio Web. Actas del Congreso Interacción 2004. Lleida
[62] G. Brajnik (2004). Comparing accessibility evaluation tools: a method for tool effectiveness. Universal Access in the Information Society, Vol 3, Nos. 3-4, Springer Verlag, pp. 252-263.
[63] D. Diaper, L. Worman (2003) Two Falls out of Three in the Automated Accessibility Assessment of World Wide Web Sites: A-Prompt v. Bobby. in Johnson, P., Palanque, P. and O'Neill, E. (Eds.) People and Computers XVII. Springer-Verlag. Pp. 349-363.
[64] Adaptive Technology Resource Centre: http://www.utoronto.ca/atrc/
[65] Access Tool Reviewer: http://www.aprompt.ca/ATR/ATR.html
[66] Developing Organizational Policies on Web Accessibility: http://www.w3.org/WAI/impl/pol.html
[67] Implementation Plan for Web Accessibility: http://www.w3.org/WAI/impl/
[68] Selecting and Using Authoring Tools for Web Accessibility: http://www.w3.org/WAI/impl/software.html
[69] Authoring Tool Conformance Evaluations: http://www.w3.org/WAI/AU/2002/tools
[70] Evaluation, Repair, and Transformation Tools for Web Content Accessibility: http://www.w3.org/WAI/ER/existingtools.html
[71] Overview. Planning Web Accessibility Training: http://www.w3.org/WAI/training/
[72] Curricula for Web Accessibility Training: http://www.w3.org/WAI/training/cr.html
[73] Evaluating Web Sites for Accessibility: http://www.w3.org/WAI/eval/
[74] Review Teams for Evaluating Web Site Accessibility: http://www.w3.org/WAI/eval/reviewteams.html
[75] Lynx: http://lynx.browser.org/
[76] Jaws for Windows: http://www.freedomscientific.com/fs_products/software_jaws.asp
[77] IBM Home Page Reader 3.04: http://www-306.ibm.com/able/solution_offerings/hpr.html
[78] J. Nielsen, R. L. Mack (1994). Usability Inspection Methods. Wiley.
[79] J. Rubin (1994). Handbook of Usability Testing". J. Wiley & Sons.
[80] Template for Accessibility Evaluation Reports: http://www.w3.org/WAI/eval/template.html
[81] J. Nielsen (2001). The usability lifecycle. http://www-106.ibm.com/developerworks/library/it-nielsen3/
[82] D. J. Mayhew (1999). The Usability Engineering Lifecycle: A practitioner's handbook for User Interface Design. Morgan Kaufman.
[83] T. Granollers (2004). MPIu+a. Una metodología que integra la ingeniería del software, la interacción persona ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares. Tesis doctoral. Universidad de Lleida.
[84] International Organization for Standardization (ISO): http://www.iso.org
[85] G. Brajnik (2001). Towards valid quality models for websites. In Proc. of 7th Human Factors and the Web Conference, Madison, Wisconsin, June.
[86] UsableNet: http://www.usablenet.com
[87] J. Thatcher, C. D. Waddell, S. Lawton Henry, S. Swierenga, M. D. Urban, M. Burks, B. Regan, P. Bohman (2002). Constructing accessible web sites. Glasshaus (UK).
[88] J. A. Jacko, A. Sears (2003). The Human-Computer Interaction Handbook, Fundamentals, Evolving Technologies and Emerging Applications. Lawrence Erlbaum Associates.
[89] Evaluation and Repair Tools Working Group: http://www.w3.org/WAI/ER/
[90] Evaluation and Report Language: http://www.w3.org/TR/EARL10/
[91] J. Darzentas, A. Arnellos, J. Darzentas, P. Koutsabasis, T. Spyrou, N. Viorres, E. Vlachogiannis, C. Velasco, Y. Mohamad, J. Abascal, J. Tomás, M. Arrue, N. Tsopelas, N. Floratos (2003). IRIS: Implementing an Open Environment supporting Inclusive Design of

Internet Applications. In: Rauterberg M., Menozzi M., Wesson J. (eds.): Human-Computer Interaction. IOS Press, pp. 749-752.

10 Patrones e interfaz de usuario

Pedro J. Molina

Imagen y Medios S.L.
C/ Cronista Carreres, 9, 8º-B,
46003 Valencia, España
pjmolina@pjmolina.com
http://pjmolina.com

10.1 Introducción

Los patrones han tomado un gran auge durante los últimos años en el desarrollo de aplicaciones. La disciplina de construcción de interfaces de usuario no ha dejado pasar esta oportunidad para aplicarlos también al área. En el ámbito de la Ingeniería del Software, los patrones se aplicaron en primer lugar dentro de la comunidad orientada a objetos.

Esta nueva forma de ingeniería trataba de buscar en la realidad, problemas comunes para abstraer la esencia del problema y de su solución. De este modo, se consigue obtener un catálogo de pares problemas-soluciones donde cada problema tiene un nombre, contextos de aparición y posibles soluciones. Aunque los problemas en ocasiones puedan tener una difícil solución, el simple hecho de documentarlos y darles nombre permite a la comunidad de desarrollo dotarse de una jerga común donde todos los que se refieren a un mismo problema usan el mismo nombre para referirse a él, constituyendo de por sí un modo nuevo de difundir el conocimiento o *know-how* del área.

El proceso de identificación de patrones requiere de grandes dosis de observación, para encontrar semejanzas y de abstracción, para descartar detalles irrelevantes y describir con precisión la esencia del problema. De este modo, el proceso de identificación de patrones puede verse como un tipo de ciencia empírica y de abstracción donde a partir de los resultados se intenta obtener leyes que explican dichos resultados.

Este origen empírico de los patrones implica que el uso de patrones constituya una ciencia de carácter estadístico ya que los patrones, por sí mismos y de modo aislado, no resuelven todos los problemas posibles. Sin embargo, una colección de patrones cuidadosamente escogidos puede caracterizar un gran porcentaje, dependiendo del dominio y su amplitud, de los pares problema/solución para un dominio dado. Es más, la decisión de la incorporación o no de un nuevo patrón a una colección de patrones puede verse justificada por la frecuencia/infrecuencia de aparición del problema descrito por dicho patrón frente al coste u oportunidad de su incorporación al lenguaje de patrones (coste/beneficio).

Como se verá más adelante, un **lenguaje de patrones** es un conjunto de patrones organizado, referidos a un mismo contexto, donde se describen las interrelaciones entre los patrones y sus posibles incompatibilidades. Los lenguajes de patrones pueden ser usados de este modo para resolver problemas de más alto nivel mediante la adopción de soluciones alcanzadas por composición de patrones del lenguaje.

10.1.1 Origen

Los patrones dentro de la comunidad informática comenzaron a tomar fuerza a partir de la relevancia obtenida por el libro *Design Patterns: Elements of Reusable Object-Oriented Software* [Gamma95] de Eric Gamma, Richard Helm, Ralph Jonson y John Vlissides (frecuentemente citados como *the Gang of Four*, la banda de los cuatro, o más frecuente todavía, por su acrónimo *GoF*). Éste libro recopila una serie de patrones de diseño orientados a objetos que ayudan a la construcción y posterior reuso de componentes de software a un nivel de diseño.

Sin embargo, el origen de los patrones es previo. El uso del termino actual *patrón* fue descrito por el arquitecto Christopher Alexander que escribió diversos libros sobre planificación urbana y construcción de edificaciones: *Notes on the Synthesis of Form* [Alexander64], *The Oregon Experiment* [Alexander75], *A Pattern Language: Towns, Buildings, Construction* [Alexander77] y *The Timeless Way of Building* [Alexander79].

Aunque los textos de Alexander tratan sobre arquitectura, las ideas que en ellos se intentan captar pueden ser aplicadas a otras disciplinas, en particular, al software, disciplina en la que por el momento han sido más aclamados que en su dominio original, la arquitectura.

Alexander argumenta que los métodos arquitectónicos actuales conducen a productos que no satisfacen las necesidades reales ni los requisitos de los usuarios. No cumplen con el propósito final de todo diseño o esfuerzo de ingeniería: mejorar la condición humana.

Alexander pretende crear estructuras que sean adecuadas para las personas, mejorando su confort y calidad de vida. Así, en *The Timeless Way of Building*, Alexander trata de captar ideas de diseño atemporales (*timeless*) para tratar de alcanzar estos objetivos. El paradigma para arquitectura propuesto por Alexander se basa en tres conceptos:

- **La cualidad** (*the Quality Without a Name*). Es la esencia de las cosas útiles que imprime en ellas cualidades como: libertad, completitud, confort, armonía, habitabilidad, durabilidad, apertura, resistencia, variabilidad y adaptabilidad. Es lo que nos hace sentir satisfechos de un producto o diseño y finalmente mejora la condición humana.
- **La puerta** (*the Gate*). Es el mecanismo que permite alcanzar *la cualidad*. Se manifiesta como un lenguaje de patrones común que permite crear múltiples diseños para satisfacer diferentes necesidades. Es el universo etéreo de los patrones y sus relaciones permitidas en un dominio dado. *La puerta* conduce a *la cualidad*.

- **El camino** (*the Timeless Way*). Usando *el camino*, los patrones definidos en *la puerta* se aplican y se combinan progresivamente para obtener diseños que poseen *la cualidad*.

Una vez introducido el concepto de patrón, merece la pena revisar algunas definiciones del concepto de diversos autores para contrastar diferentes puntos de vista.

Rhiele y Zullighoven [Riehle96] proponen una definición de patrón como sigue:

"Un patrón es la abstracción de una forma concreta que se mantiene en contextos específicos y no arbitrarios."

Los autores de esta definición apuntillan que, dentro de la comunidad informática, la noción de patrón es *guiada hacia la resolución de problemas*. Es decir, la forma concreta que se repite como solución a un problema recurrente.

Una definición más compacta que refleja este contexto es la proporcionada por Appleton [Appleton97]:

"Un patrón es una semilla nominada de información instructiva que captura la estructura esencial y la perspicacia de una familia de soluciones probadas a un problema recurrente que surge dentro de un cierto contexto entre intereses o fuerzas contrapuestas."

Los patrones también son concebidos como tipos de arquitecturas u organización de partes constituyentes para producir entidades más complejas. En este sentido, una tercera definición de patrón podría ser la propuesta por el mismo Alexander [Alexander79].

Cada patrón es una regla compuesta por tres partes, que expresan una relación entre un cierto contexto, un problema y una solución.

- Como elemento en el mundo, cada patrón es una relación entre un cierto contexto, un cierto sistema de fuerzas que ocurren repetidamente en ese contexto, y una cierta configuración espacial que permite resolver las fuerzas por sí mismas.
- Como elemento del lenguaje, un patrón es una instrucción que muestra como la configuración espacial debe ser usada, una y otra vez, para resolver el sistema de fuerzas dado siempre que el contexto sea relevante.

"El patrón es, en resumen, al mismo tiempo una entidad, que ocurre en el mundo, y una regla que nos dice como crear esa entidad, y cuando debemos crearla. Es a la vez un proceso y una entidad. La descripción de una entidad que está viva, y la descripción del proceso que genera esta entidad."

Por último, Coplien [Coplein96] propone un símil de la definición previa respecto a los patrones de costura:

"Podemos explicar cómo hacer un vestido especificando la ruta de las tijeras sobre la tela en términos de ángulos y longitudes de corte. O bien, podemos dar un patrón. Leyendo la especificación de corte, nadie tendría idea acerca de que se está construyendo, al menos hasta que esté construido. Sin embargo, el patrón presagia el resultado: es la regla para crear el vestido, pero también, en gran medida, es el vestido en sí mismo."

Como se ha podido comprobar a través de la las definiciones previas, el patrón involucra una descripción general de una solución recurrente a un problema recurrente que tiene objetivos y restricciones, *fuerzas*, definidos. Además de todo esto, un patrón hace mucho más que identificar una solución, también justifica por qué la solución es necesaria poniendo en evidencia la existencia del problema original.

10.1.2 Formatos de descripción

Existen numerosos formatos de descripción de los patrones, yendo desde el estilo narrativo puro alejandrino [Alexander77] hasta las descripciones basadas en plantillas más formalizadas [Gamma95]. El uso de plantillas está justificado para dotar de cierta estructura a una colección de patrones, sobretodo porque facilita su indexación por propiedades, semántica o relaciones ayudando a su posterior localización. El número de secciones a emplear y su semántica, sin embargo, es muy variable. Un dominio puede tener aspectos de interés que pueden justificar usar una sección en la plantilla y sin embargo no ser relevantes en otro dominio. Es por ello que habitualmente, las plantillas de definición de patrones suelen variar según el autor del patrón.

Sin embargo, parece razonable, sino de facto consensuado, al menos en la ingeniería del software y en la interacción hombre-maquina que al menos deberían aparecer las siguientes secciones un patrón:

- **Nombre:** Debe ser un nombre significativo, corto, mediante una palabra simple o una frase breve que permita referir el patrón de manera no ambigua. Si en la literatura se utilizan distintos nombres para un mismo patrón, debe hacerse constar otros nombres empleados para el mismo concepto.
- **Problema:** Una frase o párrafo que describa su intención; así como las metas y objetivos *requisitos* que se persiguen dentro del contexto y fuerzas. A menudo, las fuerzas se oponen a los objetivos o los objetivos se excluyen mutuamente de algún modo.
- **Contexto:** Las precondiciones bajo las cuales el problema y su solución son recurrentes. Por estas precondiciones la solución es la deseable. El contexto informa acerca de la *aplicabilidad* del patrón. El contexto puede ser visto como la configuración inicial del sistema antes de que el patrón sea aplicado.

- **Fuerzas:** Una descripción de las fuerzas relevantes y restricciones y como interaccionan o entran en conflicto entre sí y con los objetivos que se persiguen. Las fuerzas revelan la complejidad del problema y definen los tipos de *intercambios* que deben ser considerados en la presencia de tensiones o disonancias creadas por las fuerzas. Una buena descripción de un patrón debe reflejar todas las fuerzas que tienen impacto sobre el patrón.
- **Solución:** Relaciones estáticas y reglas dinámicas que describen como alcanzar el resultado deseado. A menudo consiste en la descripción de una serie de pasos que permiten construir el producto necesitado. La estructura estática describe la forma y organización del patrón, mientras que el comportamiento dinámico es el que da la semántica al resultado.
- **Ejemplos:** Consiste en uno o varios ejemplos de aplicaciones del patrón que ilustran: un contexto inicial, cómo se aplica el patrón y cómo transforma el contexto. Los ejemplos ayudan a entender el uso del patrón y su aplicabilidad.
- **Contexto resultante:** Describe el estado o configuración del sistema una vez el patrón ha sido aplicado incluyendo las Consecuencias, pros y contras, derivados de la aplicación del patrón. Incluye los problemas y patrones que pudieran aparecer dentro de ese nuevo contexto y describe las postcondiciones y efectos laterales del patrón. A este apartado, a menudo se le denomina *resolución de fuerzas* debido a que describe qué fuerzas han sido resueltas, cuáles permanecen sin resolver y qué patrones son nuevamente aplicables en el estado actual.
- **Fundamento:** Una explicación que justifique los pasos o reglas descritos en el patrón. Debe explicar como las fuerzas y restricciones son orquestadas para conseguir los requisitos deseados. Es la justificación de utilidad del patrón: explica cómo funciona, por qué funciona y por qué es una *buena solución*.
- **Patrones relacionados:** La relación con otros patrones si la hubiera. Los patrones dentro de un mismo lenguaje de patrones suelen tener fuerzas compartidas.
- **Usos conocidos:** Describe ocurrencias conocidas del patrón y su aplicación dentro de sistemas existentes. Esta ayuda valida un patrón verificando que efectivamente es una *solución probada* a un *problema recurrente*.

10.1.3 Lenguajes de patrones

Un lenguaje de patrones es un conjunto de patrones que comparten un mismo ámbito o contexto, es decir, comparten un mismo dominio y además están relacionados entre sí. Un lenguaje de patrones puede servir de base para un marco de trabajo o *framework*. Este marco de trabajo es reutilizable y puede ser aplicado para resolver problemas dentro de ese ámbito bien definido.

Los patrones pueden clasificarse de acuerdo con el nivel de abstracción al que pertenecen. De acuerdo con esta clasificación, y dentro del ámbito de sistemas de información existen:

- **Patrones arquitectónicos (o de análisis):** Los patrones arquitectónicos expresan estructuras de organización de sistemas de información. Proporcionan un conjunto de subsistemas predefinidos donde se especifican sus responsabilidades y se incluyen reglas y guías para organizar las relaciones entre los subsistemas. Una arquitectura de tres capas o cliente/servidor son ejemplos de patrones arquitectónicos.
- **Patrones de diseño:** Un patrón de diseño proporciona un esquema para refinar componentes o subsistemas de un sistema de información o las relaciones entre ellos. Este patrón describe una estructura recurrente de componentes que se comunican resolviendo un problema general de diseño en un contexto particular. Aunque los patrones de diseño suelen proporcionar ejemplos sobre un lenguaje de programación como C++ o Java, son independientes de lenguaje.
- **Patrones de programación, *Idioms*:** Un patrón de programación o *idiom* es un patrón de bajo nivel, muy especifico para un lenguaje de programación dado [Coplein98]. Este tipo de patrón describe como implementar aspectos particulares de componentes o sus relaciones entre ellos usando las características particulares de un lenguaje de programación dado. Es decir, un *idiom* no puede ser reusado en otro lenguaje.

Otra clasificación de patrones propuesta por Riehle [Riehle96] incluye:

- **Patrones conceptuales:** Un patrón conceptual es un patrón que se describe en términos de conceptos de un dominio de aplicación, es decir, dominio del problema.
- **Patrones de diseño:** Un patrón de diseño es aquel que se describe en términos de construcciones de diseño: v.g. objetos, clases, herencia, agregación y relaciones de uso.
- **Patrones de programación:** Un patrón de programación está descrito en términos de construcciones de un lenguaje de programación.

Los patrones están en la punta de lanza de la investigación para sistemas de información. Desde la publicación de *Design Patterns* [Gamma95] muchos otros libros, talleres (*workshops*) y conferencias se han realizado acerca de los patrones de diseño.

Para cualquier diseñador de sistemas, los patrones son una fuente de conocimiento que no debe ser despreciada, ya que contienen el conocimiento y experiencia de otros diseñadores de sistemas sobre problemas que aparecen una y otra vez.

En las siguientes secciones del presente capítulo se introducirá un formato de descripción de patrones y una relación de cualidades que debería poseer un patrón (secciones 10.2 y 10.3, respectivamente). A continuación, las secciones 10.4 y 10.5

introducen un recorrido por los patrones de interfaz de usuario desde los puntos de vista de diseño y conceptual. La sección 10.6 da cuenta de los usos que pueden hacerse de los patrones en el desarrollo de software a lo largo de sus diferentes etapas de desarrollo. Finalmente, la sección 10.7 presenta las conclusiones a este capítulo.

10.2 PLML

PLML son las siglas correspondientes a *Pattern Language Mark-up Language* (Lenguaje de marcado para lenguajes de patrones). PLML corresponde a una especificación de *mínimos* para permitir escribir patrones en un formato basado en XML que fue propuesta durante el transcurso del taller: *HCI Patterns: Concepts and Tools*, en la edición de CHI'2003 [CHI03].

La motivación para la definición de este estándar de mínimos persigue promover la adopción de un formato de intercambio entre herramientas para patrones. De este modo: herramientas de autor *para escritores de patrones*, catálogos de patrones *para búsqueda y aprendizaje* y otras herramientas adicionales podrán acceder a la información disponible en forma de patrón.

La especificación de PLML en su primera versión es muy somera y sólo aborda el aspecto estructural de los patrones. En cualquier caso es un compromiso entre su total automatización y estructuración y la libertad que demanda una especificación narrativa en lenguaje natural al estilo de los patrones de Alexander.

La Figura 1 muestra la estructura impuesta por PLML a los patrones descritos con este formato. Las secciones principales que incorpora PLML son las siguientes:

- Nombre
- Contexto
- Fuerzas
- Problema
- Solución
- Relaciones

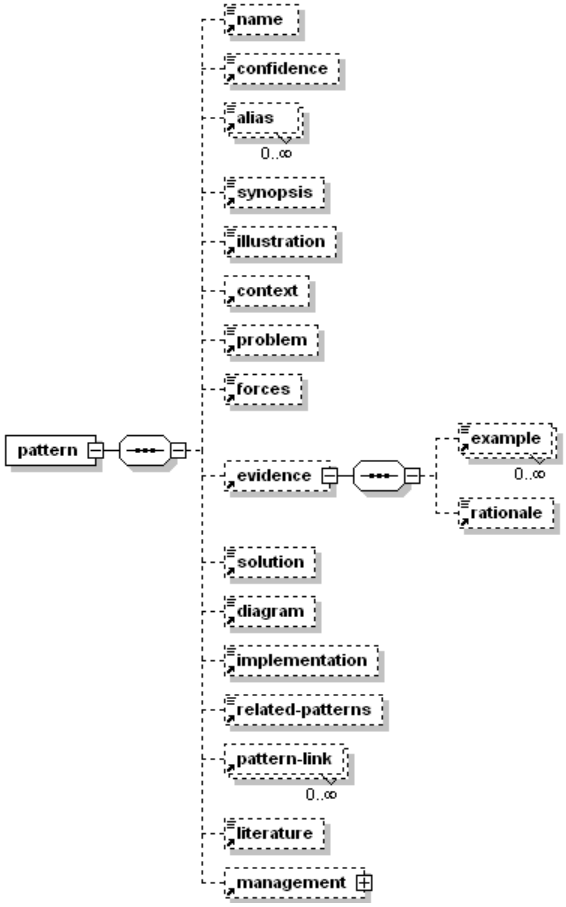


Figura 10.1. Estructura de PLML.

Existe una DTD para PLML 1.00 que indica las restricciones que deben satisfacer todo patrón expresado en PLML. La DTD con la especificación de PLML versión 1.00 puede descargarse desde:
http://www.pjmolina.com/es/investigacion/intereses.php

10.3 Cualidades de un patrón

Un patrón bien descrito debe exhibir las siguientes cualidades deseables según Lea [Lea99]:

- **Encapsulación y abstracción:** Cada patrón encapsula un problema bien definido y su solución en un dominio particular. Los patrones deben proporcionar fronteras claras que ayuden a separar claramente el espacio

del problema del espacio de la solución parcelando el patrón en fragmentos interconectados.

- **Apertura y variabilidad:** Los patrones deberían ser abiertos para soportar extensiones o parametrización por parte de otros patrones de modo que puedan trabajar juntos para resolver problemas mayores. La solución de un patrón debe ser capaz de ser realizada mediante diferentes implementaciones.
- **Generabilidad y composicionabilidad:** Un patrón una vez aplicado genera un contexto resultante que encaja con el contexto inicial de uno o más patrones en un lenguaje de patrones. Los subsiguientes patrones se aplican para, poco a poco, alcanzar la solución completa. Los patrones se aplican de modo incremental (*piecemeal growth*). La aplicación de un patrón proporciona un contexto inicial para la aplicación del siguiente patrón. (Extraído del *Call for Papers* del congreso PLoP'97*, Proceedings of Language of Patterns 1997*).
- **Equilibrio:** Por último, cada patrón debe realizar algún tipo de balanceo entre las fuerzas y restricciones involucradas. Puede deberse a que hay invariantes o heurísticos que son empleados para minimizar el conflicto en el espacio de la solución. Los invariantes a menudo tipifican problemas fundamentales que pueden ser resueltos mediante un principio de resolución para el dominio particular.

Por otro lado, Winn y Calder [Winn02] han propuesto una lista de propiedades esenciales que son observables en un patrón. Si bien esta lista de propiedades no garantizan la condición de *ser,* su ausencia proporciona indicios claros de su *desclasificación* como patrón. Estas propiedades observables son:

- **Un patrón implica un artefacto.** El patrón presagia la forma del resultado del mismo modo que el patrón de costura presagia la prenda [Coplein96].
- **Un patrón cruza varios niveles de abstracción.** Un patrón no es ni un diseño concreto, ni una idea totalmente abstracta. Al contrario, el patrón facilita la progresión de un nivel de abstracción al siguiente.
- **Un patrón es a la vez funcional y no funcional.** Los temas funcionales tratan la posibilidad y determinan las decisiones en un determinado contexto. Los temas no-funcionales se ocupan de la viabilidad: las decisiones deseables en contextos particulares. Un patrón incluye ambos tipos de consideraciones.
- **Un patrón es una solución manifiesta.** Allí donde un patrón haya sido usado consciente o inconscientemente para resolver un problema, el patrón estará presente y será reconocible en la solución diseñada.
- **Un patrón captura los puntos críticos de un sistema.** Los patrones facilitan un buen diseño capturando lo que Wolfgang Pree denomina los puntos críticos de un sistema (*system "hot spots"*). Los elementos clave son aquellas partes de una solución que pueden cambiar conforme el sistema evoluciona. Los patrones capturan los invariantes y los puntos

calientes y proporciona una estructura para manejar la interacción entre la parte estable y los elementos cambiantes del sistema.

- **Un patrón es parte de un lenguaje.** Cada patrón aparece conectado con otros patrones. Los patrones forman parte de redes de patrones interrelacionados. El mismo Alexander los concibió en origen de este modo [Alexander77].
- **Un patrón es validado por su uso.** Normalmente, los patrones son descubiertos más frecuentemente a través de experiencias concretas que a partir de disquisiciones abstractas. Sea como fuere, un patrón no puede ser verificado o validado desde un planteamiento exclusivamente teórico. La demostración de la existencia del patrón, en otras palabras, la justificación de su necesidad de ser, recae en la aparición constatable de instancias del patrón en artefactos del dominio.
- **Un patrón está vinculado a un dominio**. Un patrón no es una unidad aislada, si no que es definido el un contexto con otros patrones (lenguajes de patrones). Un patrón aplicado fuera del dominio inicial puede no ser viable.
- **Un patrón captura una gran idea.** Los patrones no son soluciones a problemas triviales. Cada solución a un problema particular no garantiza la existencia de un patrón.

En resumen, un patrón debe proporcionar un equilibrio entre el problema que identifica y las soluciones específicas que propone.

10.4 Patrones en la interfaz de usuario

Durante los últimos años, los talleres (*workshops*) organizados a lo largo de sucesivos años en el marco de la conferencia CHI (*Computer Human Interaction*) patrocinados por ACM/SIGCHI (grupo de interés en Interacción Persona Ordenador de la *Association for Computer Machinery*) han sido el principal foro para la discusión acerca de la aplicación de los patrones al desarrollo de interfaces de usuario.

En el año 1997, Thomas Erickson y John Thomas sentaron las bases para la búsqueda de un lenguaje de patrones para la interacción.

En la edición del año 2000, Richard Griffits, Lyn Pemberton, Jan Borchers y Adam Stork organizaron un segundo taller: *Pattern Languages for Interaction Design: Building Momentum*.

En la edición del año 2002, Martijn van Welie, Kevin Mullet y Paul McInerney organizaron un taller enfocado al uso práctico de los patrones [CHI02].

Durante la edición de 2003 se celebró un taller orientado hacia herramientas de soporte a *Perspectives on HCI patterns: concepts and Tools* organizado por Sally Fincher, Janet Finlay, Sharon Greene, Lauretta Jones, Paul Matchen, John Thomas y Pedro J. Molina [CHI03]. En este último, el énfasis se puso en especialmente en las herramientas de soporte a los patrones. Fué en este taller, como se ha indicado antes, de donde surgió la iniciativa de la notación PLML.

Como resultado de estos talleres, se aprecia un interés creciente en el campo de los patrones software aplicados al área de la construcción de interfaces de usuario.

Dado el diferente bagaje de los asistentes a estos talleres, se han obtenido una visión enriquecedora al contrastar el uso que se está haciendo de los patrones desde diferentes puntos de vista: v.g. como herramienta para la transmisión de conocimiento, como herramienta de diseño, como lenguaje común para los desarrolladores y usuarios o como soporte a la especificación y generación de código a partir de modelos.

10.4.1 Patrones de diseño

Jennifer Tidwell [Tidwell99] publicó una de las colecciones de patrones de diseño de referencia en interfaz de usuario. La colección de patrones de Tidwell está estructurada en las siguientes secciones: organización del contenido, acceso a la información, organización de las páginas (o formularios), entrada de datos, visualización de información compleja, comandos y acciones, manipulación directa y elementos estilísticos.

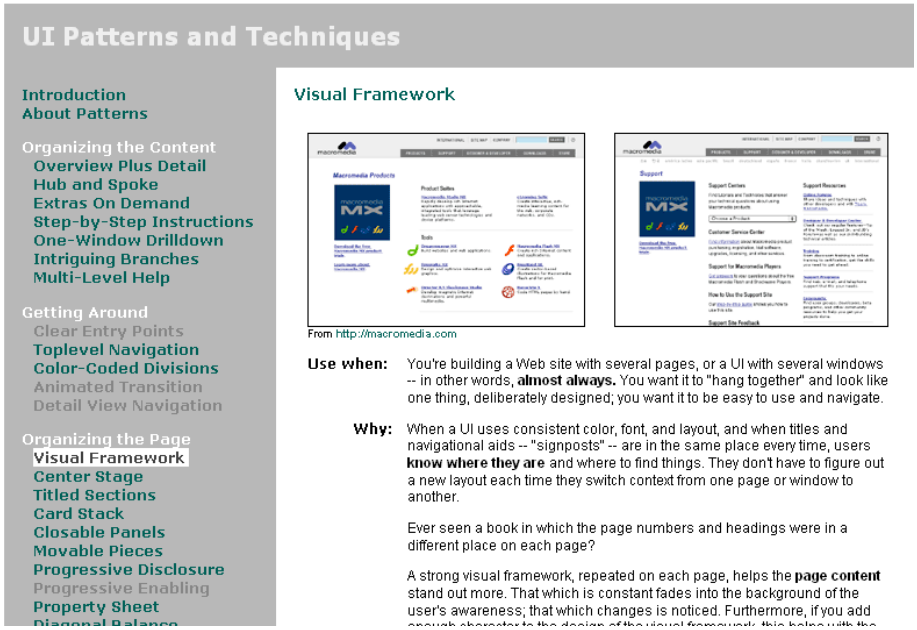


Figura 10.2. Patrón *Visual Framework* [Tidwell99].

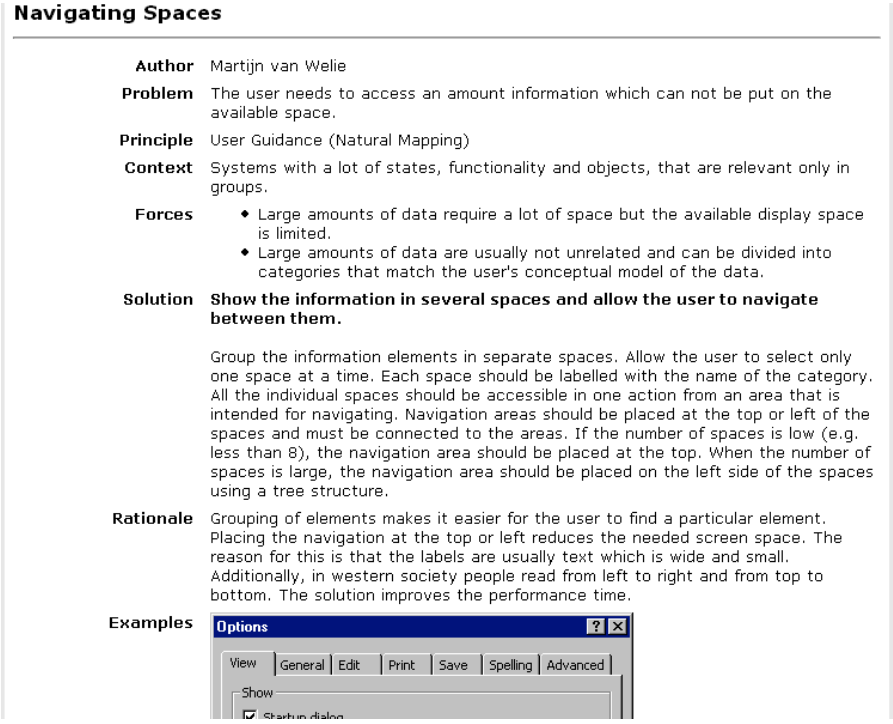
La Figura 2 muestra un ejemplo de patrón (*Visual Framework*) descrito por Tidwell. La exposición es muy clara y sencilla en un estilo muy narrativo con muy pocas secciones: un nombre, un ejemplo visual a modo de titular complementario que permite recordarlo fácilmente, cuando usarlo, por qué, cómo y ejemplos.

Los patrones de Tidwell son una fuente excelente para diseñadores noveles ya que son ideas sencillas de diseño, no triviales, que necesitan ser aprendidas por los

diseñadores. En este contexto, su utilidad didáctica es valiosísima debido a que van directamente al grano, evitando detalles superfluos.

Martijn van Welie dispone de otra colección más extensa en número de patrones organizada por dominios: diseño web, aplicaciones de escritorio y aplicaciones móviles. La Figura 10.3 muestra el patrón *Navigation Spaces* [Welie00] de esta colección. La plantilla de descripción de patrón es más elaborada y más técnica, estando más alejada de una descripción en lenguaje natural para ganar en estructuración y formalidad.

Los libros *Pattern Approach to Interaction Design* [Borchers01] o el más reciente: *The Design of Sites* [Duyne02] complementan la bibliografía de referencia sobre patrones de diseño en interfaz de usuario, este último libro especialmente orientado hacia el diseño de sitios web.



Navigating Spaces

Author Martijn van Welie

Problem The user needs to access an amount information which can not be put on the available space.

Principle User Guidance (Natural Mapping)

Context Systems with a lot of states, functionality and objects, that are relevant only in groups.

Forces
- Large amounts of data require a lot of space but the available display space is limited.
- Large amounts of data are usually not unrelated and can be divided into categories that match the user's conceptual model of the data.

Solution **Show the information in several spaces and allow the user to navigate between them.**

Group the information elements in separate spaces. Allow the user to select only one space at a time. Each space should be labelled with the name of the category. All the individual spaces should be accessible in one action from an area that is intended for navigating. Navigation areas should be placed at the top or left of the spaces and must be connected to the areas. If the number of spaces is low (e.g. less than 8), the navigation area should be placed at the top. When the number of spaces is large, the navigation area should be placed on the left side of the spaces using a tree structure.

Rationale Grouping of elements makes it easier for the user to find a particular element. Placing the navigation at the top or left reduces the needed screen space. The reason for this is that the labels are usually text which is wide and small. Additionally, in western society people read from left to right and from top to bottom. The solution improves the performance time.

Examples

Figura 10.3. Patrón *Navigation Spaces* [Welie00].

10.4.2 Patrones de análisis / conceptuales

Además de los populares patrones de diseño, es posible encontrar patrones de mayor nivel de abstracción útiles para las etapas de toma de requisitos o de análisis.

En el libro *Patrones de Análisis* [Fowler96] se describen una serie de patrones característicos del análisis de sistemas de información. Abarcando campos como la contabilidad, monitorización, almacenaje y facturación, por citar algunos, se describen patrones de análisis para capturar la funcionalidad asociada a esos sistemas que reaparecen una y otra vez en los sistemas de información.

Por otro lado, en el ámbito de la interfaz de usuario también se han explorado este tipo de patrones llamados de análisis o conceptuales (v.g. Just-UI [Molina02]) en los que se trata de capturar las propiedades de la interfaz de usuario a nivel conceptual, mucho antes de tomar en consideración la plataforma final de implantación, y previo al diseño de la interfaz.

10.4.3 Clasificación de Patrones

Dependiendo de su dominio y nivel de abstracción, podemos situar las diferentes colecciones y lenguajes de patrones. En particular, la Figura 10.4 muestra una clasificación teniendo en cuenta las fases de desarrollo y su grado de abstracción respecto del dominio considerado incluyendo la funcionalidad de la aplicación o los aspectos de interfaz.

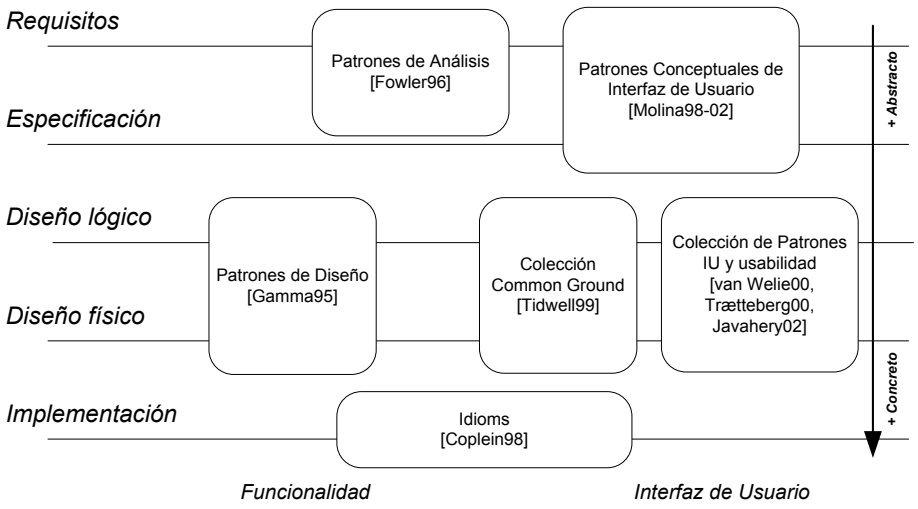


Figura 10.4. Comparación de colecciones de patrones.

Los *Patrones de Diseño* [Gamma95] son patrones de propósito general, por lo que no están restringidos a interfaces de usuario, que pueden ser aplicados para resolver problemas de diseño. Por contra, los *Patrones de Análisis* [Fowler96] son considerados en las fases de requisitos y especificación. Fowler proporciona diversos patrones para diferentes dominios como son la banca, la sanidad o la contabilidad. Dentro del mundo de las interfaces de usuario, las colecciones de patrones suelen enfocarse a la fase de diseño [Tidwell99, Welie00, Traetteberg02, Javahery02].

10.5 Patrones Conceptuales de Interfaz de Usuario

Los patrones de interfaz de usuario a nivel conceptual han sido explorados para la especificación de interfaces de usuario independientemente del dispositivo o canal [Molina02].

En esta línea, Just-UI [Molina03a] es un modelo para la especificación abstracta de interfaces de usuario. El modelo se ha desarrollado usando un lenguaje de patrones obtenido de modo empírico sobre el dominio de aplicaciones de gestión. Además, se han desarrollado herramientas de especificación y de generación de código completa para diversas plataformas a partir de este modelo (OlivaNova Modeler [CARE03]).

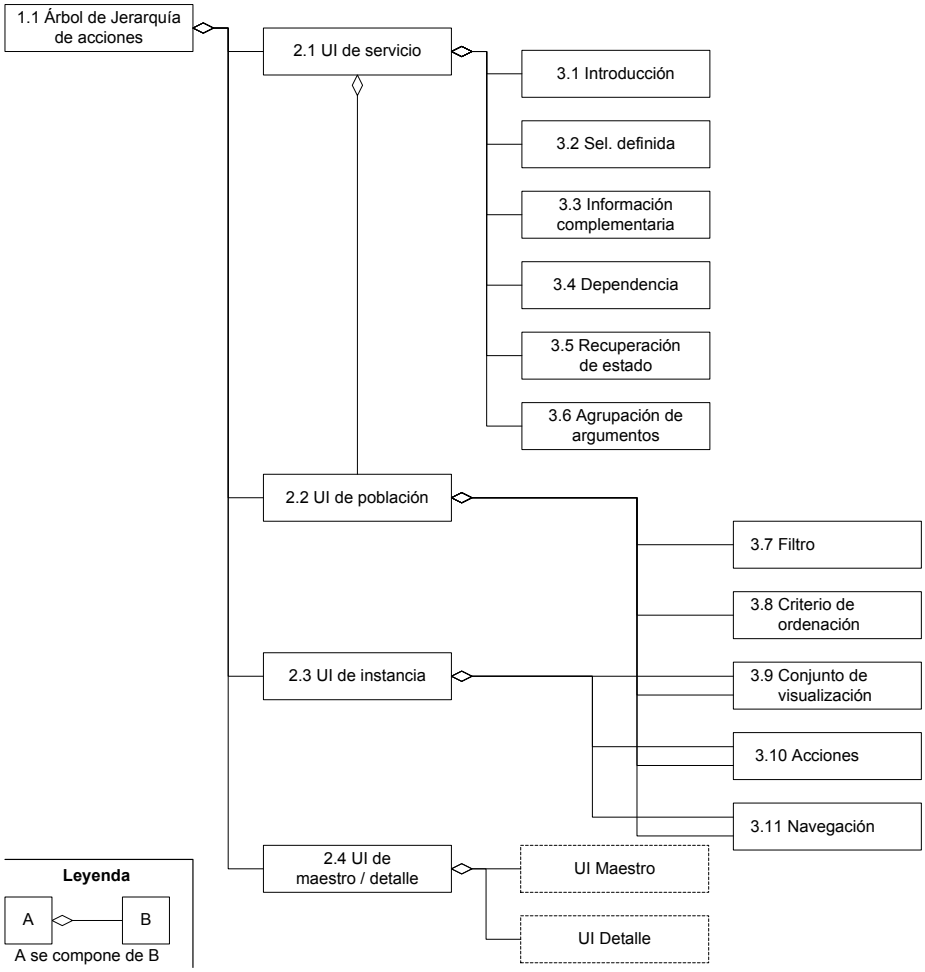


Figura 10.5. Lenguaje de Patrones de Just-UI.

Just-UI se apoya en un método orientado a objetos para dar cuenta de las entidades y sus relaciones en el espacio del dominio. La información recogida de este modo, es

empleada para la construcción de interfaces de usuario de modo manual o de modo automatizado con la ayuda de generadores de código diseñados al efecto para diversos lenguajes de implementación.

La Figura 10.5 ilustra el lenguaje de patrones propuesto en Just-UI con sus relaciones de uso. El lenguaje se estructura en tres niveles de abstracción:

1. **Acceso a la información:** compuesto por un solo patrón denominado *Árbol de Jerarquía de Acciones*.
2. **Unidades de interacción:** cuatro patrones que dan cuenta de configuraciones básicas de interfaces de usuario para aplicaciones de gestión.
3. **Patrones elementales:** compuesto por once patrones básicos que decoran o complementan a los patrones de segundo nivel.

Como muestra de patrones conceptuales de interfaz de usuario se describirá uno por cada nivel. Para más detalle de descripción del lenguaje consulte [Molina03b].

1. Jerarquía de acciones
2. Población
3. Filtro

10.5.1 Nivel 1. Árbol de Jerarquía de Acciones.

El nivel 1 está constituido por un solo patrón: el Árbol de Jerarquía de Acciones (AJA) [Molina03b]. Este patrón se emplea para definir el acceso a la funcionalidad de la aplicación por parte del usuario. Dependiendo del tipo de usuario, los permisos establecidos, las tareas que deba llevar a cabo o el dispositivo a emplear, el acceso a presentar puede ser radicalmente distinto. Por este motivo, se sugiere disponer de un análisis previo de tareas u otra descomposición basada en tareas como ayuda al diseño de AJA efectivos.

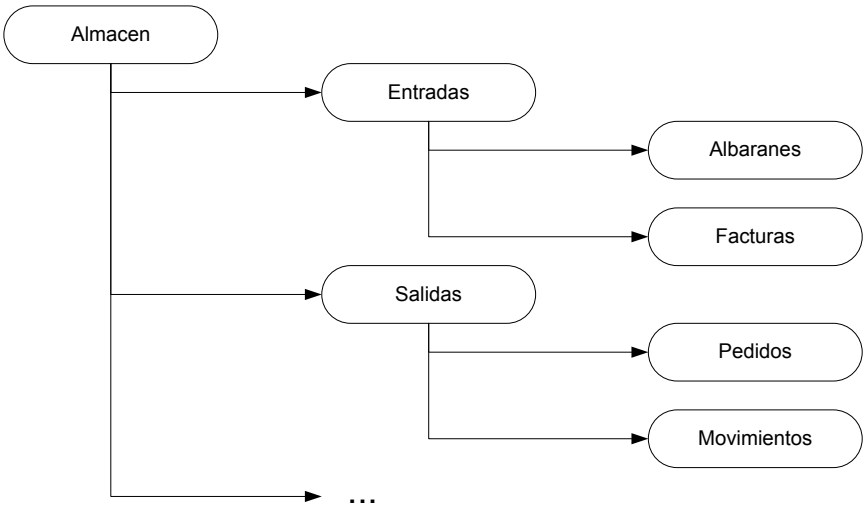


Figura 10.6. Ejemplo de Árbol de Jerarquía de Acciones.

La Figura 10.6 muestra un ejemplo de patrón de jerarquía de acciones donde se organiza la funcionalidad que será ofrecida al usuario en un determinado sistema. Algunas implementaciones efectivas de este patrón son:

- un menú para una aplicación en Windows, o bien,
- una serie de enlaces o un árbol dinámico en *Javascript* para una página web.

10.5.2 Nivel 2. Unidad de Interacción de Población.

Como muestra de patrón de nivel 2 (unidades de presentación) se ha seleccionado el patrón Población [Molina03b]. Su uso está orientado a mostrar y trabajar con colecciones de objetos de un mismo tipo (v.g. objetos instancias de una misma clase).

Para su especificación hay que indicar: la clase a la que queremos hacer referencia y una serie de subpatrones de nivel 3 que complementan o decoran la semántica de esta unidad de interacción:

1. **Filtros:** ¿cómo la información debe ser filtrada?
2. **Conjuntos de visualización:** ¿qué campos son relevantes para el usuario?
3. **Criterio de ordenación:** ¿cómo debe ser ordenada la información?
4. **Acciones:** ¿qué acciones son relevantes para el usuario?
5. **Navegación:** ¿qué información relacionada debería estar accesible?

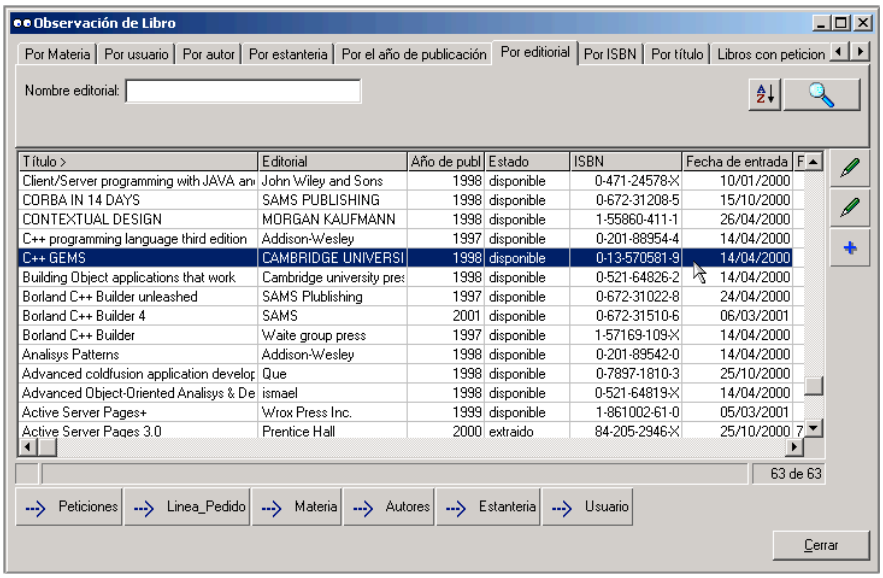


Figura 10.7. Ejemplo de implementación de Población en Windows.

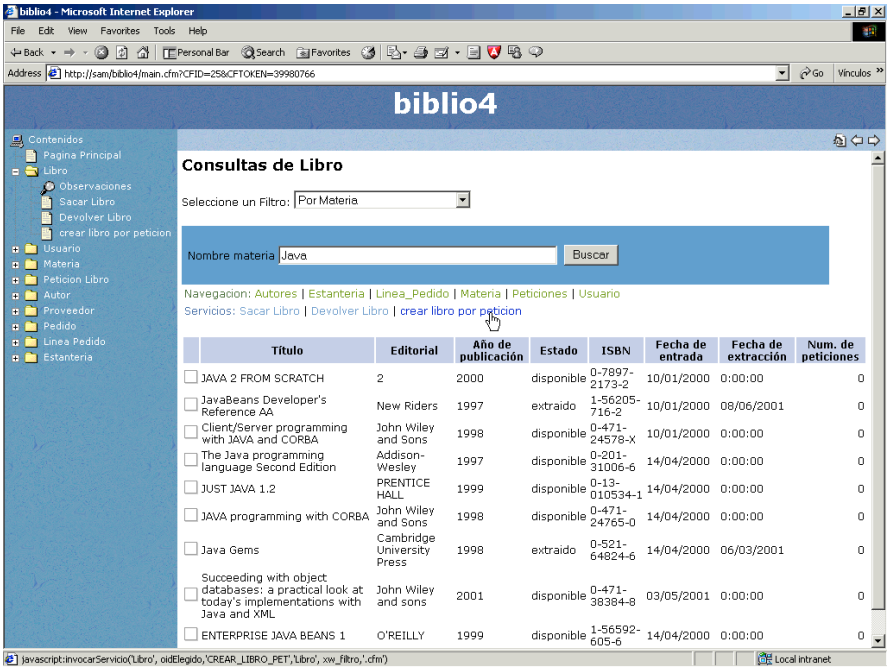


Figura 10.8. Ejemplo de implementación de Población en Web.

La Figura 10.7 y la Figura 10.8 muestran dos ejemplos de implementación del patrón Población: como formulario basado en una rejilla en Windows, y como una tabla para web, respectivamente.

La especificación de la necesidad del usuario ha quedado capturada y reflejada en el modelo por medio de la instanciación del patrón sin comprometer todavía ningún detalle de diseño. Esté modo de proceder, es el que ha permitido que tanto la interfaz de la Figura 10.7 como la de la Figura 10.8 hayan podido ser completamente generadas a partir de la información del modelo conceptual.

10.5.3 Nivel 3. Filtro

Como ejemplo de nivel 3, el Filtro [Molina03b] es un patrón que permite restringir la población de objetos para obtener sólo aquellos en los que el usuario está interesado. Se expresa como una formula bien formada en términos de las propiedades de los objetos a filtrar con el objetivo final de trabajar sólo con aquellos que satisfacen la condición de filtro.

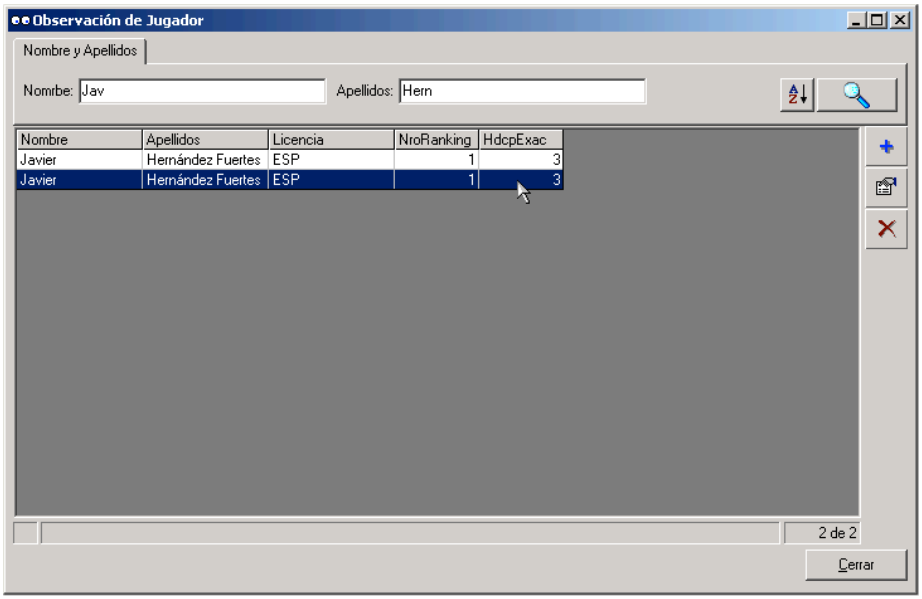


Figura 10.9. Ejemplo de filtro.

La Figura 10.9 muestra un ejemplo de implementación de filtro en Windows. El control de pestañas (o *tab control*) de la parte superior implementa el filtro contenido dentro de un patrón población. El filtro habilita un mecanismo de búsqueda de personas por nombre y apellidos.

El patrón filtro, no puede ser aplicado en cualquier situación. Por ello el lenguaje de patrones restringe su uso proporcionando reglas de composición: para este caso, el patrón Filtro solo puede aplicarse en el contexto de un patrón Población. Es decir, allí donde puedan aparecer colecciones de objetos que necesiten mecanismos de búsqueda.

10.5.4 Notación gráfica y elicitación de requisitos.

Asociado a Just-UI, se definió [Molina03b] una notación gráfica que permite plasmar los conceptos de modelado mediante un diagrama según los niveles (véase Figura 10.10).

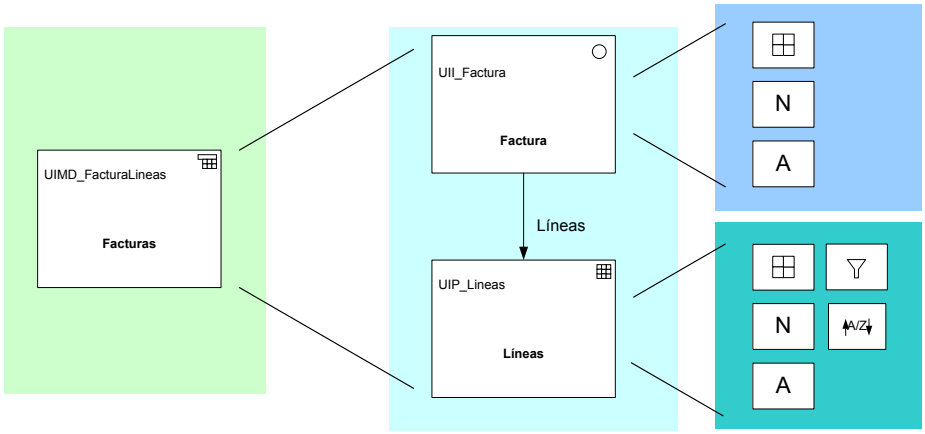


Figura 10.10. Despiece de la notación gráfica.

Está capacidad de reflejar gráficamente los patrones y sus composiciones en un momento dado habilitan su uso para elicitación de requisitos y validación del mapa navegacional de la interfaz de usuario.

Al mismo tiempo, la notación gráfica hace al modelo más escalable. El método puede ser usado en sistemas más complejos, ya que la notación gráfica permite descomponer el problema en partes o subsistemas mucho más manejables.

10.6 Usos de los patrones

Los patrones sirven de ayuda en las diferentes fases de la concepción de software. Erickson apuntó que una colección de patrones proporciona una jerga sobre un dominio que constituye de por sí una *lingua franca* [Erickson99]. Todo el equipo humano involucrado en el desarrollo: usuarios, analistas, diseñadores, programadores, creativos pueden usar esta jerga específica proporcionada por los patrones para entender y transmitir el conocimiento del problema y de la solución propuesta a los demás. Este es por tanto el primer, y tal vez más importante, uso de los patrones, es decir, favorecen la comunicación en el equipo de desarrollo.

En la presente sección, se describirán algunas aplicaciones para cada fase del desarrollo de un producto software.

10.6.1 Toma de requisitos y análisis

Durante el proceso de toma de requisitos analistas y clientes tienen una fuerte interacción para conseguir clarificar los requisitos de sistema a construir. La importancia de esta fase es vital, puesto que de la buena o mala compresión que se alcance de los requisitos, determinará en gran medida la calidad de la aplicación entendida como la adecuación del software a la función a la que deben dar soporte.

En esta fase, los patrones de análisis (para dar cuenta de la funcionalidad de la aplicación [Fowler96] o la interfaz de usuario [Molina03a]) ayudan a los analistas a identificar necesidades. Un patrón de análisis puede ajustarse a un determinado requisito. En tal caso, el patrón seleccionado proporciona un contexto de aplicación que sirve para acotar los casos donde el patrón es efectivo. El patrón aplicado de este modo deja en el sistema una nueva configuración, aporta estructura y semántica, y sus implicaciones han de ser sopesadas en su justa medida, en especial respecto a otros requisitos del sistema con los que podría entrar en conflicto.

A su vez, los patrones de análisis, al estar expresados en términos del espacio del problema, son cercanos y entendibles con poco esfuerzo por el cliente. Constituyen de este modo una herramienta más para el arsenal del analista a la hora de comunicarse con el cliente. El análisis del sistema descrito de este modo, soportado con patrones, puede ser validado por parte del usuario.

10.6.2 Diseño

La fase de diseño es la primera que empezó a sacar partido a los patrones. Las colecciones de Jennifer Tidwell [Tidwell99] o Martijn van Welie [Welie00] aportan soluciones de diseño reusables que concentran la experiencia de diseñadores expertos y que pueden ser aplicadas de nuevo en situaciones o problemas similares.

10.6.3 Implementación

A nivel de implementación los patrones se denominan *Idioms* [Coplein98]. Normalmente, son dependientes del lenguaje de programación. Pueden agruparse a modo de *librerías de funciones* que implementan funcionalidad bien conocida, efectiva y probada.

Aunque no muy extendidas con este nombre, las librerías, *frameworks* o componentes (COTS) pueden verse en ocasiones como formas de agrupar patrones de implementación para su posterior reuso.

10.6.4 Generación de código

De un modo simplista, podríamos ver los patrones como pares problema-solución (o conjunto de soluciones). Así pues, los catálogos de patrones pueden almacenar también implementaciones reusables de los problemas que describen.

Si un generador de código o un compilador es capaz de producir una aplicación a partir de una especificación o entrada, un generador de código puede igualmente leer una especificación basada en patrones y transformar los patrones identificados en soluciones. Más aún, si la especificación del patrón se realiza de un modo lo suficientemente formal, en teoría una aplicación automatizada podría seleccionar el mejor patrón que se ajusta para un problema dado usando técnicas de inteligencia artificial o técnicas heurísticas.

Ejemplos de estos usos de los patrones han sido experimentados sobre los patrones de diseño [Budinski96] y sobre interfaz de usuario [Molina03b].

10.6.5 Mantenimiento

El impacto de los patrones en el código es bastante marcado. Más aún si cabe cuando son relativos a interfaz de usuario donde la mayoría se ponen de manifiesto por la mera manipulación de la interfaz de usuario. Este rastro dejado en la solución final por los patrones, favorece la trazabilidad: de código a diseño para patrones de diseño y de código a análisis para patrones conceptuales o de análisis.

Este modo de proceder hace más legible y mantenible el software: el por qué de la elección de un determinado patrón responde a un determinado problema de diseño o de análisis.

La *lingua franca* [Erickson00] en la fase de mantenimiento vuelve a cobrar importancia. El equipo de desarrollo puede volver a retomar el análisis y diseño de la aplicación y apoyarse en la jerga proporcionada por los patrones para entender y transmitir los cambios que se necesiten realizar para adaptar la aplicación a la nueva realidad.

10.6.6 El efecto Delta

Los patrones pueden ser aplicados en cadena. Propuestas de uso como *Pattern Supported Approach* [Grandlund99] han sido presentadas en esta línea. Esta forma de usar o aplicar los patrones como puentes entre los distintos niveles de abstracción ayuda a reducir el *gap semántico* entre niveles.

Cuanto más abstracto es un patrón, más amplio es el posible campo de aplicación. Al mismo tiempo, el patrón puede ser instanciado para resolver el problema con múltiples formatos. A raíz de esta observación, las siguientes preguntas surgen de modo natural en fases de diseño:

- *¿Qué implicaciones impone la elección de un determinado patrón en la fase de análisis?*

O más específicamente:

- *¿Cómo los patrones conceptuales aplicados en la fase de análisis influyen en las fases siguientes?*

Un patrón proporciona restricciones semánticas, de comportamiento y estructurales. Dichas restricciones tienen un alcance cónico o Efecto Δ (Delta) [Molina02] (véase Figura 10.11): la elección tiene implicaciones desde su definición hasta los refinamientos en las sucesivas fases. Estos refinamientos están, por tanto, supeditados a las decisiones tomadas en fases precedentes.

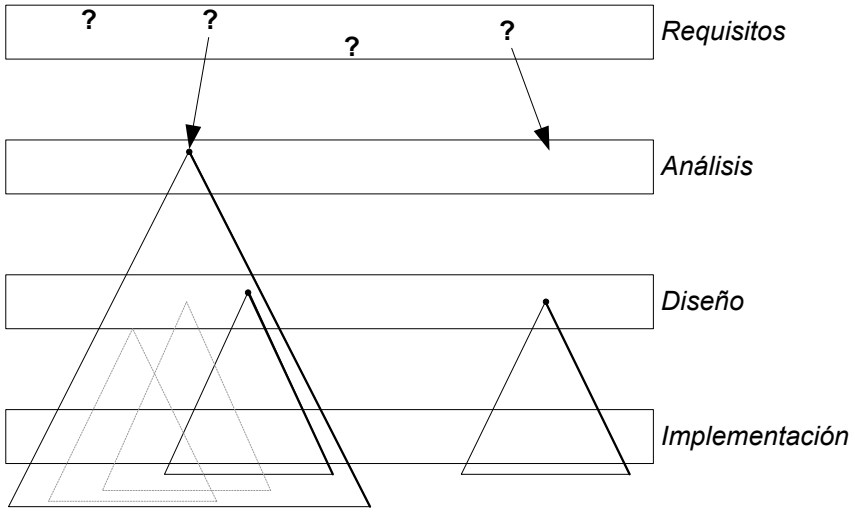


Figura 10.11. Efecto Delta (Δ).

Los símbolos de interrogación mostrados en la Figura 10.11 representan preguntas o requisitos formulados en la fase de requisitos. Dichas preguntas son respondidas por medio de la elección de un patrón conceptual en la fase de análisis. La Figura 10.11 muestra el alcance de las decisiones en forma de triángulos Δ donde aspectos de subsiguientes fases están restringidos o guiados por decisiones previas.

En este sentido, decisiones tomadas en las primeras fases (selección de patrones conceptuales en la fase de análisis) pueden guiar o ayudar a tomar decisiones en términos de diseño (donde los patrones de diseño pueden ser aplicados). Una herramienta de tipo asistente (*wizard*) puede ayudar a seleccionar los patrones en la fase de diseño a partir de la información capturada en las fases previas. Por ejemplo, un asistente puede sugerir la implementación del Patrón *Maestro/Detalle* [Molina02] en la fase de diseño mediante los patrones de diseño *Container Navigation* [Traetteberg02] o *Navigation Spaces* [Welie00].

De este modo, los patrones son resueltos en cascada. Los patrones conceptuales pueden ser implementados usando patrones de diseño y estos, a su vez, aplicando *idioms*.

El Efecto Δ (Delta) así observado, permite restringir y guiar la selección de opciones de diseño en una fase de construcción en función de las elecciones tomadas en una fase previa. De este modo, se puede dar soporte a un refinado guiado de las especificaciones hasta la implementación final.

10.7 Conclusiones del capítulo

En este capítulo se ha presentado el origen e impacto de los patrones en el mundo del software, y más particularmente, aplicados al mundo de la interfaz de usuario. El impacto y algunos de los posibles usos de los patrones en las diferentes etapas de desarrollo de software han sido presentados.

Diversos esfuerzos se han venido aplicando para intentar incrementar el nivel de especificación formal de los patrones. Aunque complicado de alcanzar, por la propia naturaleza empírica de los patrones, una mayor formalización permitirá en el futuro cercano:

- la catalogación y creación de librerías de patrones más extensas a partir de la agregación de colecciones de diferentes autores individuales,
- incrementar el corpus de conocimiento experto sobre contextos de uso especializados, como pueda ser, la interfaz de usuario,
- la aparición de herramientas automatizadas para: escribir, explorar, sugerir, aplicar y generar código y detectar patrones, entre otros usos.

Precisamente al respecto, el en capítulo 12, Susana Montero profundiza en este tema y propone una catalogación de patrones para el dominio de hipermedia para integrar los patrones en el proceso de diseño.

10.8 Referencias

[Alexander64] C. Alexander. *Notes on the Synthesis of Form*. Harvard University Press, 1964.

[Alexander75] C. Alexander. *The Oregon Experiment*. Oxford University Press, 1975.

[Alexander77] C. Alexander. A *Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[Alexander79] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[Appleton97] B. Appleton. *Patterns and Software: Essential Concepts and Terminology,* 1997. http://www.enteract.com/~bradapp/docs/patterns-intro.html

[Borchers01] J. Borchers, *A Pattern Approach to Interaction Design*, John Wiley & Sons, 2001.

[Budinski96] Budinsky, Finnie, Vlissides, and Yu. *Automatic code generation from design patterns*. IBM System Journal, 1996. http://www.research.ibm.com/journal/sj/352/budinsky.html

[CARE03] CARE Technologies S.A. *OlivaNova Software* 2003. http://www.care-t.com

[CHI02] M. van Welie, K. Mullet y P. McInerney. *CHI2002 Patterns Workshop*, 2002. http://www.welie.com/patterns/chi2002-workshop/

[CHI03] S. Fincher, J. Finlay, S. Greene, L. Jones, P. Matchen, P.J. Molina y J. Thomas. *CHI 2003 Workshop. Perspectives on HCI patterns: concepts and tools*, 2003. Computer Human Interaction 2003 SIGCHI. Fort-Lauderdale, Florida, EE.UU. 2003. http://www.dsic.upv.es/~pjmolina/CHI2003WS/

[Coplein96] J.O. Coplein. *Software Patterns*. SIGS Books & Multimedia, Nueva York, EE.UU., 1996.

[Coplein98] J.O. Coplein. *C++ Idioms*. En Proceedings of the EuroPlop'98., 1998. http://www.bell-labs.com/user/cope/Patterns/C++Idioms/EuroPLoP98.html

[Duyne02] Douglas K. van Duyne, James A. Landay, Jason I. Hong. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience.* Addison-Wesley Professional, 2002.

[Erickson00] T. Erickson. *Lingua francas for design: sacred places and pattern languages*. In Proceedigns of the Conference on Designing Interactive Systems. pagínas 357-368. ACM Press, 2000.

[Fowler96] M. Fowler. *Analysis Patterns: Reusable Object Models.* Object-Oriented Software Engineering Series. Addison-Wesley, Reading, MA, EE.UU. 1996.

[Gamma95] E. Gamma, R. Helm, R. Johnson y J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading MA, Addison Wesley Professional Computing Series 1995.

[Grandlund99] A. Granlund, D. Lafreniere. *A Pattern Supported Approach to User Interface Design*. Workshop in UPA Conference. 1999. http://www.upassoc.org/conf99reg/ws6.shtml

[Javahery02] H. Javahery y A. Seffah. *A Model for Usability Pattern-Oriented Design*. Pribeanu y Vanderdonckt (eds.), TAMODIA'200, pags. 104–110. Bucarest, Rumania, Julio 2002.

[Lea99] D. Lea. *Christopher Alexander: an introduction for Object-Oriented Designers*, 1999. http://www.enteract.com/~bradapp/docs/patterns-intro.html

[Molina02] P.J. Molina, S. Meliá y Ó. Pastor. *User Interface Conceptual Patterns*. En P. Forbrig, Q. Limbourg, B. Urban y J.Vanderdonckt (Editores), Design, Specification, and Verification of Interactive Systems, págs. 201–214. Université catholique de Louvaine, University of Rostock, Louvain-La-Neuve, Belgica y Rostock, Alemania, Junio 2002. También en Proceedings of the 4th International Workshop on Design Specification & Verification of Information Systems DSV-IS'2002, Springer Verlag, Berlin, Alemania, Diciembre 2002. ISBN 3-540-00266-9. http://www.springer.de/cgi/svcat/search_book.pl?isbn=3-540-00266-9

[Molina03a] P.J. Molina, S. Meliá y Ó. Pastor. *The Just-UI approach: Conceptual modelling of device independent interfaces*. En Christophe Kolski y Jean Vanderdonckt (Editor), «Journal of Human-Computer Interaction

(RIHM). Special Issue on Computer-Aided Design of User Interfaces», 1. 2003.

[Molina03b] P.J. Molina. *Especificación de interfaz de usuario: de los requisitos a la generación de código.* Tesis doctoral. Universidad Politécnica de Valencia. 2003.
Disponible en http://pjmolina.com/es/investigacion/tesis.php

[Riehle96] D. Riehle y H. Züllighoven. *Understanding and Using Patterns in Software Development*, Theory and Practice of Object Systems, 2(1): 3-13, 1996.

[Tidwell99] J. Tidwell. *Common Ground: A Pattern Language for Human-Computer Interface Design*, 1999.
http://www.mit.edu/~jtidwell/common_ground.html

[Trætteberg02] H. Trætteberg. Model-Based User Interface Design. Tesis Doctoral, Norwegian Uiversity of Science and Technology, Trondheim, Noruega, Mayo 2002.
http://www.idi.ntnu.no/~hal/publications/thesis/thesis.pdf

[Welie00] M. van Welie. «The Amsterdam Collection of Patterns in User Interface Design», 2000. http://www.cs.vu.nl/~martijn/patterns/index.html

[Winn02] T. Winn y P. Calder. *Is This a Pattern?* IEEE Software, págs. 59–65, Enero/Febrero 2002.

11 CAPITALIZANDO EXPERIENCIA DE DISEÑO EN APLICACIONES WEB

Matías Butti, Juan Danculovic, Gustavo Rossi[1]
LIFIA-Facultad de Informática, UNLP
E-mail: [mbutti, jdancu, Gustavo]@sol.info.unlp.edu.ar

11.1 Introducción

La construcción de aplicaciones web de cierta complejidad involucra diferentes aspectos tanto en el nivel de diseño como de implementación. Por un lado, por el solo hecho de ser piezas de software "heredan" muchos de los problemas ya conocidos en nuestra disciplina: la inestabilidad de los requerimientos, la necesidad de modularizar correctamente las diferentes componentes para aumentar la posibilidad de reuso y simplificar el mantenimiento, etc.
Al mismo tiempo, las características intrínsecas de una aplicación Web, tales como el hecho de ser un tipo especial de aplicación hipermedia complican aún más la situación. Realizar una aplicación web no es una tarea trivial e involucra el desarrollo de varias actividades. A continuación se menciona solo una lista de las actividades más conocidas y reportadas en la literatura:

-Construir un modelo del dominio que refleje la semántica de los datos y las operaciones en la aplicación.
-Decidir que objetos del dominio serán perceptibles para el usuario, que información el mismo podrá acceder y como se organizarán estos objetos.
-Establecer las relaciones hipermediales que permitan la navegación a través del espacio de información
-Definir como se organizará la interfaz de la aplicación, de que manera el usuario interactuará con la misma, que tipo de realimentación recibirá al realizar sus acciones, etc.

Está claro, y ha sido largamente mencionado en la literatura [Batory 00], que aún para aplicaciones pequeñas es imposible abarcar todas las incumbencias de diseño sin una estrategia sistemática; en ese sentido la mejor y mas conocida forma de hacerlo es "divide y vencerás", más precisamente separar las incumbencias de diseño con precisión, construir modelos que de alguna manera resuelvan los problemas en cada una de ellas y encontrar mecanismos claros y no ambiguos para integrar dichos modelos.
Nuestra experiencia con OOHDM [Schwabe 98] (Object-Oriented Hypermedia Design Method) y otros métodos como UWE [Hennicker 00] y WebML [Ceri 00] nos permiten afirmar que existen ciertas actividades que ya se consideran estándar en el dominio web: el modelado conceptual o aplicativo, el diseño de la navegación, el diseño de la interfaz gráfica y la implementación. Para cada una de ellas los distintos métodos proponen ciertas primitivas de modelado que permiten al diseñador expresar con cierta precisión los elementos del modelo (ejemplos interesantes son los presentados en los capítulos 2, 3, 4 y 5 de este libro).

[1] Gustavo Rossi es parcialmente apoyado por la UAI en su proyecto: "Modelizacion Conceptual de Aplicaciones Web"

Sin embargo conocer un método, o sea sus primitivas de modelado y el proceso de desarrollo asociado, no nos garantiza poder resolver los problemas que debemos enfrentar en cada aplicación. Muchas veces dichos problemas requieren algo más que conocimiento de un método, requieren experiencia, capacidad de ver más allá de los elementos básicos de la metodología; es aquí donde los patrones, en particular los patrones hipermedia, juegan un rol fundamental.
En este capítulo reflejamos brevemente nuestra experiencia de minería de patrones asociados al dominio web e hipermedia; en particular mostramos de qué manera podemos utilizar ciertos patrones en la construcción de una aplicación web. La organización del capítulo es la siguiente: primero describimos brevemente la base conceptual sobre la cual se ha basado nuestro trabajo, el método OOHDM. Seguidamente introducimos un problema sencillo, la construcción de un campus virtual e indicamos algunos de los problemas asociados a su diseño. Luego, describimos un conjunto de patrones que hemos descubierto en los últimos años, mostrando de qué manera la aplicación de dichos patrones permite resolver los problemas descritos previamente. Presentamos finalmente algunas conclusiones y temas que consideramos importante profundizar.

11.2 Diseño de aplicaciones Web

Las ventajas de construir aplicaciones web usando un enfoque centrado en modelos han sido discutidas en otros capítulos de este libro [2, 3, 4 y 5]. En particular OOHDM está basado en una clara separación de incumbencias de diseño, concretamente: diseño conceptual o aplicativo, diseño de la navegación y diseño de la interfaz.
Utilizando el método OOHDM una aplicación hipermedia se construye a través de un proceso que consta de cuatro actividades que se detallarán mas adelante. OOHDM soporta tanto un modelo de proceso incremental como un modelo de proceso basado en prototipos.
Cada paso se centra en un aspecto de diseño particular. Se utiliza la Clasificación, agregación y generalización/especialización durante todo el proceso para mejorar el poder de abstracción y las oportunidades de reutilización.
A continuación se presenta un resumen de la metodología OOHDM. El proceso consiste en cuatro etapas para una posterior implementación. Las etapas son: elicitación de requisitos, diseño conceptual, diseño navegacional, diseño de interfaz. En este capítulo, por cuestiones didácticas y de espacio nos enfocaremos en el diseño navegacional y de interfaz.

Elicitación de requerimientos

El primer paso es reunir los requerimientos de los interesados. Para conseguir estos requerimientos, es necesario como primer paso identificar los actores (interesados) y las tareas que estos deben realizar sobre el sistema. Para cada una de las tareas descubiertas, se generan los escenarios correspondientes (secuencia de interacciones). A partir de estos escenarios se conformarán los casos de usos, que serán representados usando diagramas de interacción. Estos diagramas proveen una representación gráfica de la interacción entre el usuario y el sistema durante la ejecución de una tarea.
Los diagramas de interacción se validan con los actores y en caso de ser necesario habrá un rediseño de los mismos. Se aplican un conjunto de guías sobre estos diseños para extraer el modelo conceptual.

Diseño Conceptual

En esta etapa se construye un modelo conceptual del dominio de la aplicación en construcción utilizando los principios del modelado orientado a objetos, tal cual aparecen en UML

Diseño Navegacional

En esta etapa se describe la estructura de la aplicación hipermedia en términos de contextos navegacionales que se inducen a partir de clases navegacionales. Las clases navegacionales pueden ser nodos, enlaces, índices o tour guiados. Los contextos y las clases navegacionales, tienen en cuenta tanto los tipos de usuarios como las tareas que los mismos deben realizar.
Los nodos en OOHDM representan vistas lógicas sobre las clases conceptuales definidas durante el análisis del dominio. Para un mismo esquema conceptual se pueden definir varios modelos navegacionales, que permiten expresar diferentes vistas sobre el mismo dominio.
Los enlaces se derivan a partir de las relaciones conceptuales definidas en la etapa 1. El hecho de definir la semántica de la navegación en términos de nodos y enlaces, permite modelar el movimiento en el espacio navegacional (el subconjunto de los nodos con los cuales los usuarios pueden interactuar en cualquier momento) independientemente del modelo conceptual. Se concluye, que el modelo navegacional puede desarrollarse independientemente del modelo conceptual simplificando el mantenimiento.

Diseño de la interfaz
El modelo de interfaz se construye definiendo objetos perceptibles (v.g. una imagen, el mapa de una ciudad, etc) en términos de clases de interfaces. Las clases de interfaces se definen como agregaciones o clases primitivas (v.g. campos de texto, botones) y recursivamente otras clases de interfaces. Los objetos de interfaz mapean con objetos navegacionales, proveyendo la apariencia de los mismos. El comportamiento de la interfaz se declara especificando:

- como manejar eventos externos y generados por los usuarios
- como es la comunicación entre la interfaz y los objetos navegacionales.

Por último la implementación mapea los objetos de interfaz en objetos de implementación y pueden desarrollarse arquitecturas elaboradas, por ejemplo N-capas, donde las aplicaciones clientes comparten una capa de negocio donde residen los objetos conceptuales. La naturaleza orientada a objetos de OOHDM favorece la utilización de arquitecturas de implementación estándar como J2EE entre otras.

Parte de nuestra experiencia en la construcción de aplicaciones OOHDM ha consistido en reflexionar sobre aquellas estructuras de diseño navegacional que hacen que una aplicación sea exitosa, mas allá del método con el cual fue descrita. De esta manera y durante algunos años [Lyardet 98, Lyardet 99, Rossi 00] descubrimos una cantidad importante de patrones para aplicaciones Web. El esquema de separación de incumbencias de diseño de OOHDM nos permitió

clasificar a estos patrones en primer término en tres categorías relacionadas con la actividad de OOHDM en la cual se aplica el patrón, conceptuales, de navegación y de interfaz. A nivel conceptual es natural que un diseñador utilice los patrones de diseño conocidos en la literatura [Gamma 95], mientras que a nivel navegacional y de interfaz surgen ciertos patrones específicos. Al mismo tiempo descubrimos que existen otros ejes temáticos sobre los cuales es posible encontrar patrones: algunos de ellos tienen que ver con la naturaleza de la aplicación (v.g. comercio electrónico, enseñanza), otros con ciertos aspectos más específicos (v.g personalización) o con funcionalidades particulares (v.g búsqueda). Dichos patrones han sido documentados y pueden accederse por ejemplo en [HPat 02].
Estos patrones pueden usarse como un complemento a las heurísticas provistas por el método [Rossi 99a], y como su existencia y descripción no dependen de una notación particular, pueden también utilizarse con otros enfoques de diseño como los mencionados en este libro.
Como una ejemplificación de cual es la manera en que los patrones permiten resolver ciertos problemas particulares que aparecen en aplicaciones web, seguidamente describimos una aplicación arquetípica que usaremos como ejemplo en este capítulo.

11.3 Desarrollando *UniVirtual*, un campus universitario.

`UniVirtual` es un portal hipermedia que ofrece servicios a los estudiantes, aspirantes y profesores para complementar el desarrollo de las actividades que se pueden realizar dentro de la Universidad Nacional de La Plata. Dichas actividades pueden categorizarse en: académicas, culturales, comerciales o de ocio.
Los servicios que ofrece *UniVirtual* facilitan el desarrollo de ciertas actividades pues brindan la posibilidad de llevarlas a cabo con independencia de horario y lugar. Por ejemplo no se necesita ir a una cartelera para conocer cierta información, ni tampoco es necesario hacerlo en el horario en el que la facultad está abierta.

Teniendo en cuenta la categorización de las actividades, podemos distinguir distintos tipos de servicios:

- Servicios de información cultural.
- Servicios de información académica.
- Servicios de información comercial: ventas electrónicas de material universitario, subastas entre los integrantes de la universidad.
- Servicios de información de ocio: como lo son actividades deportivas, recitales, películas, obras teatrales, etc.

UniVirtual administra gran cantidad de información. Se debe considerar también, que gran parte de esta información se actualiza frecuentemente. Una característica interesante de esta aplicación es el hecho de ser intensiva en información lo cual la hace representativa de un dominio mucho más amplio de aplicaciones similares (v.g portales del estilo de yahoo.com, servicios de información, etc.). Al margen de las dificultades que un diseñador pueda tener para describir el modelo del dominio y la navegación sobre los objetos de interés, existe un conjunto de problemas de alto nivel que merecen ser detallados:

- Cómo conseguir que el usuario se mueva de un tipo de servicio a otro ágilmente
- Cómo lograr que el usuario se entere fácilmente de las novedades (v.g. nuevos servicios), incluso sin necesidad de explorar la aplicación
- Dado que la información tiene muchísimas categorías, cómo lograr que la estructura del portal se adecue a los intereses del usuario
- Cómo optimizar los procesos de búsqueda. De que manera se facilita la comprensión de las respuestas a una búsqueda.
- Cómo orientar al usuario cuando la navegación no es trivial.

Sin pretender con esta lista agotar los problemas que se pueden encontrar en la construcción de la aplicación, los mismos aparecen recurrentemente con independencia del dominio aplicativo (obsérvese que en su descripción no se menciona *Univirtual* ni sus aspectos específicos). Una tesis de este capítulo es que conociendo una buena variedad de patrones web, es posible encontrar soluciones de buena calidad a estos problemas.
Seguidamente describimos algunos de los patrones web más representativos y mostramos a continuación como su instanciación en *Univirtual* permite resolver los problemas mencionados.

11.4 Patrones para desarrollo de aplicaciones web

La relevancia que tiene encapsular experiencia de diseño usando patrones ya ha sido ampliamente discutida en la literatura. Como un ejemplo de cómo ciertos requerimientos pueden transformarse en patrones, tomemos el siguiente de la aplicación *Univirtual*: "El campus desea mantener informado a sus alumnos de los eventos culturales, académicos y deportivos". Este es un problema recurrente en aplicaciones web que puede resolverse usando el patrón *Novedades*, que presentaremos mas adelante en este capítulo.
En las subsecciones que siguen, presentamos algunos patrones para aplicaciones web; el formato básico de presentación contiene la intención del patrón, el problema que resuelve, una descripción de la solución y algunos ejemplos de uso en aplicaciones populares. Para cada patrón cuando su área de incumbencia está definida con precisión lo indicamos después de su nombre (v.g. navegación o interfaz). Algunos patrones tienen varias áreas de incumbencia dentro del diseño por esa razón les asignaremos el área *Miscelánea*

11.4.1 Landmark (Navegación)

Intención

Resaltar los sectores de información más relevantes de la aplicación para que el usuario pueda navegar a estos sectores de manera fácil y ágil.

Problema

En muchos casos necesitamos destacar sectores de nuestra aplicación con el fin de brindarle al usuario un fácil acceso a los mismos. Muchas veces estos sectores no se encuentran relacionados entre sí como es el caso de las tiendas virtuales que tienen muchas subtiendas. ¿Cómo hacemos que el usuario pueda navegar a estos sectores de

información en todo momento sin comprometer la estructura de navegación de nuestra aplicación?

Solución

Definir una serie de *landmarks* y proveer fácil acceso desde cualquier nodo dentro de la estructura de navegación. Los enlaces hacia los *landmarks* deben verse uniformes cuando son presentados al usuario a fin de brindarle características visuales consistentes.

Ejemplos

Pueden encontrarse ejemplos de *landmarks* en numerosas aplicaciones web. Por ejemplo en el sitio de subastas electrónicas http://www.deremate.com encontramos los *landmarks* para proveer acceso a las diferentes secciones que provee el sitio como se observa en la Figura 11.1.

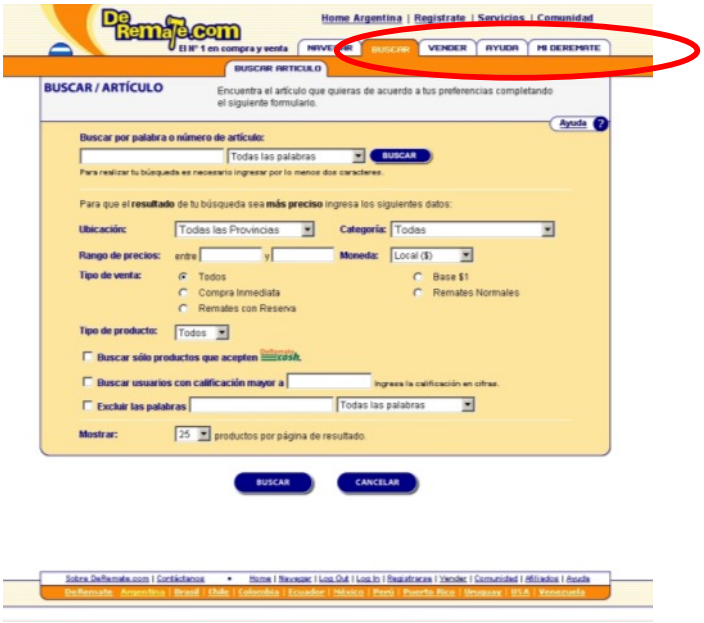


Figura 11.1: Landmarks en www.deremate.com

Otro sitio donde pueden encontrarse *landmarks* es el sitio de venta de electrodomésticos y productos electrónicos http://www.fravega.com.ar donde se utilizan para mostrar diferentes tipos de productos como se ve en la Figura 11.2.

Figura 11.2: Landmarks en www.fravega.com.ar

11.4.2 Novedades (Navegación)

Intención

Comunicar al usuario que nueva información se agregó o se modificó en el sitio web.

Problema

La información manejada por las aplicaciones web tiende a crecer. Supongamos que desarrollamos un sitio web donde ofrecemos una gran variedad de productos de hardware (v.g. impresoras, monitores, etc.). ¿Cómo podemos asegurarnos que los usuarios se enteran de los nuevos productos y servicios que se van incorporando en la aplicación?

Desde el punto de vista comercial, sería de vital importancia evitar que el usuario tenga que descubrir por su cuenta la incorporación de productos y servicios ya que se corre el riesgo que nunca lo haga.

Siguiendo las buenas prácticas de diseño, la información incorporada debería aparecer en el nodo correspondiente según los caminos de navegación, de manera que se pueda mantener el sitio bien estructurado. Los usuarios la descubrirán a través de algún camino de navegación o a través de herramientas de búsqueda. Pero esta estrategia no concuerda con la necesidad de notificar a los usuarios sobre la incorporación de información.

Entonces, ¿cómo abastecemos al usuario con información de cambios recientes o nueva información disponible sin dejar de lado las buenas prácticas para diseñar la estructura del sitio web?

Solución

Una solución elegante consiste en estructurar la página inicial de manera tal que posea una sección claramente destinada a la nueva información agregada. Para evitar mostrar la información completa en la página inicial, dicha información se presenta al usuario a través de "títulos" que se refieren a las novedades. Estos títulos que resumen la información del ítem agregado, pueden ser enlaces que nos llevan a la página que contiene la información completa.

Esta organización permite al diseñador preservar la información del sitio web organizada de una manera adecuada y a su vez notifica al usuario la actualización de información que se produce en el sitio web.
Si consideramos un sitio web estructurado en forma de árbol, la aplicación del patrón *Novedades* provee al usuario un acceso directo a la información ubicada en las hojas del árbol sin necesidad de realizar pasos de navegación intermedios con riesgo a desorientarse o directamente abandonar el proceso de búsqueda.
Los usuarios podrán encontrar la información tanto a través de los accesos directos o navegando el sitio de la manera usual.
Para presentar la información de una manera más amigable, a medida que van surgiendo nuevos títulos, los más antiguos pueden ser reemplazados o eliminados.

Ejemplos

En la figura 11.3 mostramos el uso del patrón Novedades en http://www.elcorteingles.es utilizado para mostrar las novedades de libros. Los sitios web de las empresas lo usan para presentar novedades corporativas, como por ejemplo el sitio web de la tecnología Java (http://www.java.sun.com).

Figura 11.3: Novedades en www.elcorteingles.es

También se puede encontrar el uso de este patrón en los sitios de noticias. En este caso, se aplica para dar acceso directo a las últimas (o las mas relevantes) historias como por ejemplo el periódico El mundo (http://www.elmundo.es).

11.4.3 Recomendación (Comercio Electrónico)

Intención

Ayudar al usuario a encontrar un ítem dentro de un sitio de comercio electrónico.

Problema

Muchas veces los usuarios ingresan a un sitio de comercio electrónico para ver las características de un producto que están decididos a comprar. En la mayoría de los sitios

de comercio electrónico existe gran cantidad y variedad de productos; proveer al usuario de índices sobre estos productos o motores de búsqueda dentro del sitio es sólo una solución parcial al momento de brindarle al usuario ayuda para encontrar el producto deseado.
Si aplicamos el patrón *Novedades* descrito en la subsección 11.4.2 podemos mostrarle al usuario nuevos productos o novedades del sitio. Sin embargo, las novedades pueden cambiar con frecuencia y no estamos seguros si al usuario le van a interesar los productos mostrados como novedades.

Solución

La aplicación debería dar soporte de sugerencias sobre los productos. Esta funcionalidad puede ser implementada de diferentes maneras. Por ejemplo en Amazon (http://www.amazon.com) se utiliza la historia de compras del usuario para recomendarle determinados productos al usuario. Otra manera de implementar este patrón, es utilizar una solución más generalizada, y no tan personalizada, recomendando al usuario los productos más vendidos, productos de oferta, etc. El diseño de la funcionalidad para realizar sugerencias no debe interferir con la estructura navegacional global del sitio.

Ejemplos

Nuevamente puede verse en la Figura 11.3 de El Corte Inglés una serie de indicadores de sugerencias, tales como la columna con los libros más vendidos.

11.4.4 Comunicación Push (Miscelánea)

Intención

Optimizar el proceso de búsqueda sobre áreas o productos de interés del usuario.

Problema

La interacción de los usuarios con aplicaciones web sigue un modelo *pull*, es decir, son los usuarios que recuperan la información navegando por la aplicación.
Si bien este modelo es adecuado, existen situaciones en las cuales el usuario está forzado a realizar una interacción excesiva con la aplicación para realizar la tarea que desea. Supongamos que el usuario se encuentra interesado en libros de un cierto género o planea organizar un viaje de vacaciones hacia un determinado destino. ¿Cómo podemos simplificar el proceso de búsqueda de información relacionada con estos temas? La solución convencional es que el usuario ingrese al sitio día tras día para ver si se encuentra publicada en el sitio la información que desea. Si utilizamos el patrón *Novedades* explicado en la subsección 11.4.1**,** el usuario puede enterarse de la aparición de nuevos productos o servicios, pero si aquello de su interés dejo de ser una novedad (pues no se ha conectado en días) quizás se pierda la oportunidad. Al mismo tiempo las novedades pueden abarcar muchas áreas que no son de interés del usuario.

Solución

Combinar el modelo usual de aplicaciones web basadas en un modelo *pull* con un modelo *push* para comunicación de información. Encontrar maneras de comunicarle al usuario información relevante sin obligarle que la encuentre por si mismo.
Esta solución puede implementarse de diferentes formas: una de ellas es permitir que el usuario pueda suscribirse a una determinada categoría de información (o tipo de productos), y recibir un correo electrónico cada vez que un nuevo ítem acorde a su interés aparece. Este correo electrónico puede contener una URL con un enlace para que el usuario pueda acceder de una manera *push* a la información.

Ejemplos

Numerosas variantes para este patrón pueden encontrarse en la web. Por ejemplo existe gran variedad de sitios donde el usuario configura alertas de noticias o novedades que quiere recibir seleccionando el medio por el cual recibir la misma.
Un ejemplo de este patrón lo podemos encontrar en http://www.lanacion.com.ar en el que el usuario configura las noticias que quiere recibir a su dirección de correo electrónico. Se muestra la suscripción en la Figura 11.4.

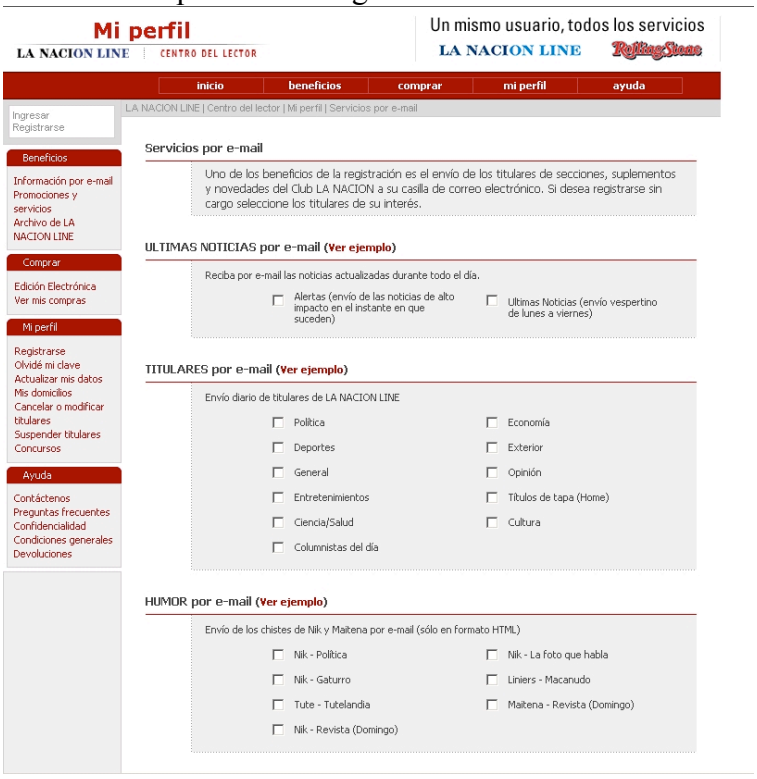


Figura 11.4 : Comunicación Push en www.lanacion.com.ar

11.4.5 Personalización de estructura

Intención

Limitar el tamaño del espacio de navegación de acuerdo con las preferencias del usuario

Problema

Algunos tipos de aplicaciones web como los portales involucran una gran variedad de categorías de información y servicios.
Cuando el usuario navega sitios como http://www.cnn.com o http://www.icq.com puede sentirse desorientado, no sólo por la cantidad de enlaces por los cuales navegar sino también por la diversidad de categorías de información y por la gran cantidad de posibles tareas a realizar. Obsérvese por ejemplo la Figura 11.5.

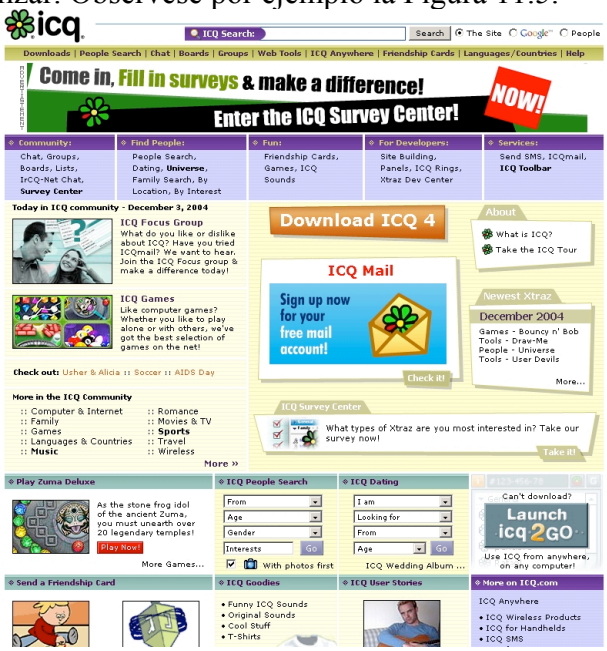


Figura 11.5: Estructura compleja en www.icq.com

Una posible solución a este problema es realizar una taxonomía de las categorías de información disponibles. El desafío está en como encontrar una estructura para los portales que brinden al usuario buenas posibilidades de navegación sin causarle una sobrecarga de información al momento de interpretar la gran cantidad de opciones disponibles que tiene para realizar.

Solución

Personalizar la estructura del sitio web o permitir que el usuario lo haga. Esto se logra proveyendo al usuario un conjunto de módulos, donde cada uno de éstos puede estar compuesto de otros módulos y/o enlaces a los objetos concretos que contienen la información. La aplicación debe brindarle al usuario la posibilidad de seleccionar sólo los módulos de información en los cuales está interesado y para cada uno de estos módulos sólo mostrar la información que el usuario quiere percibir simplificando la apariencia de las páginas web.

Ejemplos

El ejemplo más conocido de P*ersonalización de Estructura* lo encontramos en los sitios del tipo *my.xx.com* tal como www.my.yahoo.com.ar donde el usuario personaliza la página de acuerdo a sus preferencias. Por ejemplo en MyYahoo el usuario puede seleccionar un conjunto de módulos a partir de una gran variedad de los mismos que incluyen información como clima, noticias, portafolios de inversiones, salud, viajes, etc.

Cada uno de estos módulos puede ser configurado de acuerdo con las necesidades del usuario; por ejemplo éste puede escoger el clima de las ciudades del mundo que desea visualizar, noticias sobre un género musical, como mostramos en las Figuras 11.6 y 11.7.

Figura 11.6 : Selección de categorías de información en www.my.yahoo.com.ar

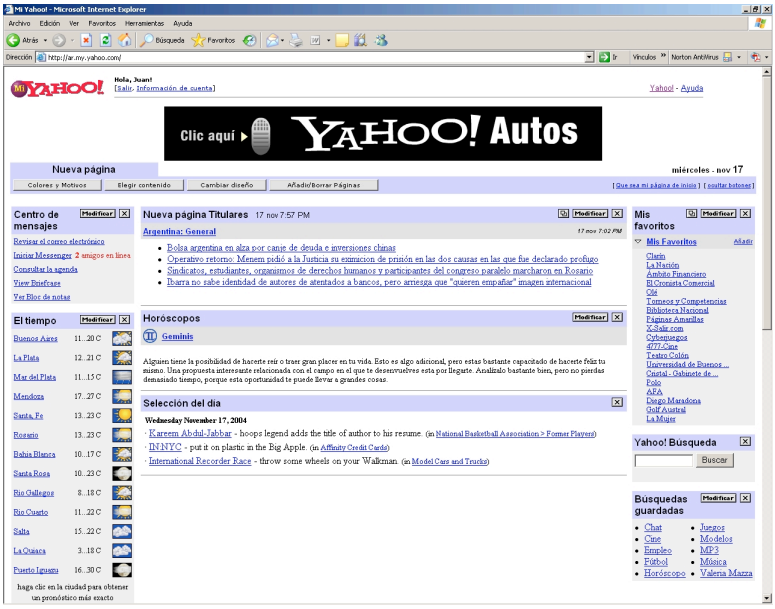


Figura 11.7: Personalización de estructura en www.my.yahoo.com.ar

11.4.6 Espacio de Búsqueda (Búsqueda)

Intención

Restringir las búsquedas a una categoría de información específica.

Problema

A medida que los espacios de información crecen, las probabilidades de encontrar la información deseada solamente mediante navegación disminuye. Es así que la mayoría de las aplicaciones provee buscadores internos para facilitar el proceso.
Cuando el usuario realice una búsqueda habrá cierta información dentro de la aplicación que será relevante para éste. Una parte de esta información relevante será útil, otra parte le será de una utilidad parcial y otra que no tendrá ningún sentido para el usuario.
Los motores de búsqueda retornarán sólo información relevante de acuerdo con los datos especificados en la consulta. ¿De qué manera podemos optimizar una búsqueda de manera que se mejore la calidad de los resultados?

Solución

Proveer al usuario de un mecanismo de selección de categorías de información (subespacios) que acote el espacio de búsqueda. Existen dos variantes para esta solución de acuerdo a la funcionalidad deseada:

- Permitir al usuario seleccionar sólo una categoría por vez. Brindamos al usuario un mecanismo de elección de subespacios disjuntos. Cada ítem de información pertenece a solo un subespacio de información por vez.
- Permitir al usuario combinar espacios de búsqueda. Un grupo de cajas de selección (*check-boxes*) es presentado al usuario con todas las posibles áreas de búsqueda. El usuario puede seleccionar cualquier combinación de ellas. Esta solución es más flexible para el usuario pero presenta un grado más de complejidad al momento de implementarla.

Ejemplos

En http://www.elcorteingles.es encontramos un ejemplo de Espacio de búsqueda implementado con una caja múltiple (*combo-box*) al momento de seleccionar sólo una categoría de búsqueda como se muestra en la Figura 11.8.

Figura 11.8: Especificación de espacio de búsqueda en www.elcorteingles.es

Otro ejemplo de Espacio de búsqueda lo encontramos en el sitio web del periódico Clarín (http://www.clarin.com.ar) en el que se puede combinar los espacios de búsqueda pudiendo combinar categorías en el momento de realizar búsquedas como se observa en la Figura 11.9.

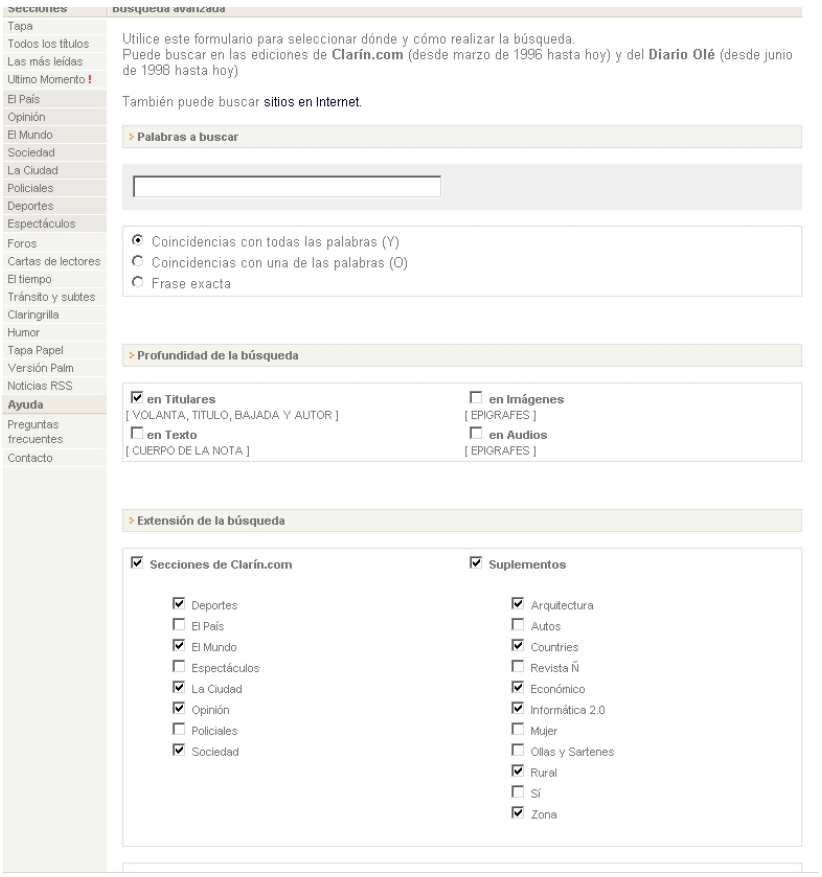


Figura 11.9: Espacios de búsqueda en www.clarin.com.ar

11.4.7 Respuesta Estructurada (Búsqueda)

Intención

Brindarle al usuario la información organizada al momento de presentarle el resultado de una consulta

Problema

En aplicaciones que administran mucha información, los resultados de las búsquedas incluyen una gran cantidad de enlaces hacia ítems que pueden ser relevantes para el usuario.
Muchas veces una simple lista de enlaces no es suficiente para cumplir el objetivo del usuario de encontrar un cierto ítem de información. Es importante complementar este resultado de manera de facilitar al usuario la selección de los ítemes deseados.

Solución

Brindarle al usuario el resultado de la consulta en forma estructurada de modo tal que la información se presente organizada de una manera que le ayude a escoger la opción más adecuada a sus preferencias de búsqueda. La información adicional puede ser agrupada dentro de diferentes categorías:

- Información del ítem encontrado: puede incluir un breve resumen de la página encontrada realizado automáticamente por el motor de búsqueda.

- Información de enlaces a otra información relacionada que es escogida de manera automática por el motor de búsqueda.
- Una taxonomía que organiza el espacio de búsqueda en subespacios.

Ejemplos

Al realizar búsquedas en http://www.elcorteingles.es podemos encontrar implementaciones del patrón *Respuesta Estructurada* al buscar por una palabra clave como se muestra en la Figura 11.10. Otra implementación interesante puede encontrarse en http://www.elmundo.es en el que el resultado de la búsqueda es organizado por categorías utilizando solapas como ilustramos en la Figura 11.11.

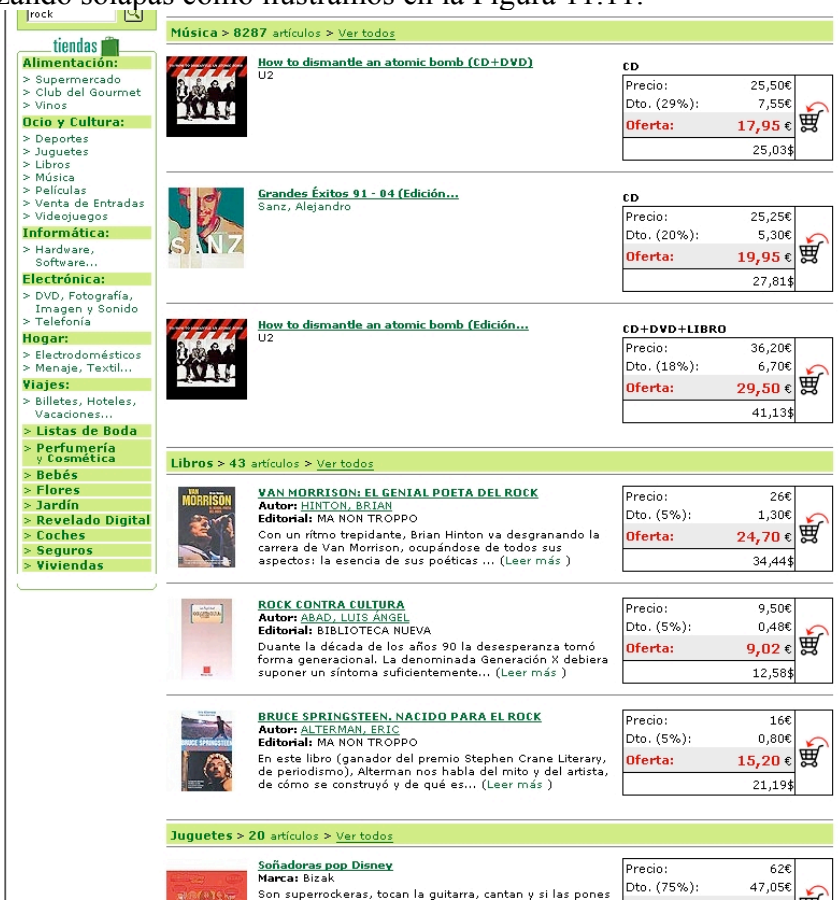


Figura 11.10: Respuesta Estructurada en www.elcorteingles.com.ar

Figura 11.11: Respuesta Estructurada en elmundo.es

11.4.8 Enlace Tipado (Usabilidad)

Intención

Ayudar al usuario a distinguir diferentes tipos de enlaces.

Problema

Presentarle al usuario enlaces en el formato convencional puede causar que ciertos enlaces de importancia sean ignorados. El origen de un enlace debería indicar que encontraremos cuando naveguemos siguiendo el enlace. Además se debe considerar que ciertos enlaces pueden indicar también archivos a ser descargados. Este último punto puede agregar otro nivel más de complejidad dentro de la apariencia visual de un sitio web, en el que el enlace puede apuntar a la sección de un documento (enlaces relativos), un documento en el mismo sitio web, un documento que se encuentra fuera de los limites del sitio web, o la descarga de archivos de diversos tipos (v.g documentos plantillas, etc.).

Solución

Clasificar los enlaces según el tipo de objeto al que apunta y asignarle un determinado icono a fin de brindarle al usuario un elemento visual adicional de información.

Ejemplos

Podemos encontrar un ejemplo del patrón de Enlace Tipado en la biblioteca digital del portal http://www.educ.ar en el que el gráfico anterior al enlace nos indica el tipo de archivo en el cual se encuentra el libro a ser consultado por el usuario como se ve en la Figura 11.12.

Figura 11.12: Link tipado en www.educ.ar

11.4.9 Información bajo demanda (Interfaz)

Intención

Organizar gran cantidad de ítems en la interfaz cuando el espacio de visualización es reducido.

Problema

Constantemente necesitamos decidir como mostrar información de un ítem navegacional. Desafortunadamente, el espacio de una pantalla suele ser reducido (lo cual puede empeorar en los dispositivos móviles) y generalmente usar otros medios, como sonido, no es adecuado o ni siquiera posible

Solución

Presentar en la interfaz sólo un subconjunto de los objetos más importantes, y brindarle al usuario la posibilidad de que controle que información será presentada. Esta acción puede realizarse a través de algún mecanismo de selección como por ejemplo botones adicionales. La selección no implica un paso de navegación sino que simplemente produce un cambio en la interfaz.

Ejemplos

Podemos encontrar un ejemplo de *Información bajo Demanda* en la sección de pronóstico del tiempo del http://www.clarin.com.ar. Cuando el usuario selecciona en un caja múltiple (*combo-box*) una nueva ciudad, la información del pronóstico cambia de acuerdo a la ciudad seleccionada sin producir un paso de navegación extra (ver Figura 11.13 y Figura 11.14).

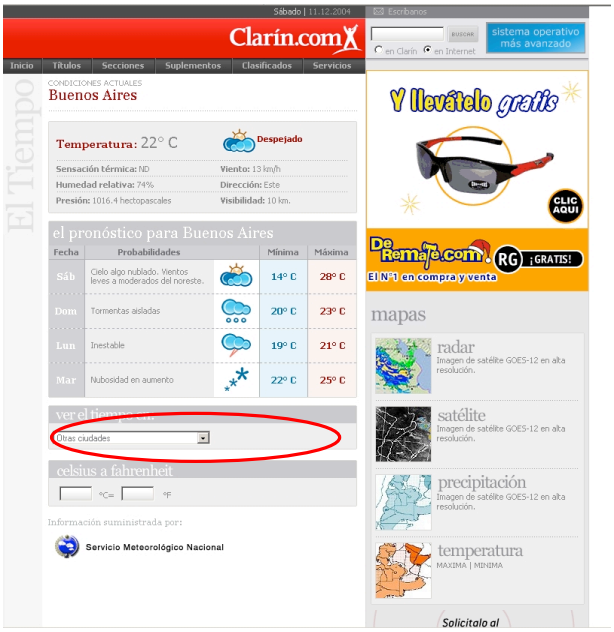


Figura 11.13: Pronostico del tiempo de buenos aires en www.clarin.com

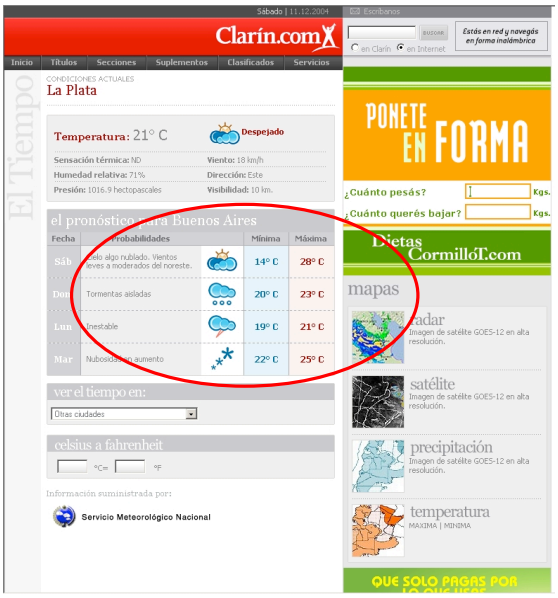


Figura 11.14: pronostico del tiempo de la plata en www.clarin.com

11.5 Resolución de los problemas de diseño de *UniVirtual* aplicando patrones.

Como un ejemplo práctico que demuestre de que manera los patrones descriptos pueden ayudarnos en la construcción de aplicaciones web complejas, mostramos seguidamente como los hemos utilizado en el proceso de construcción de *UniVirtual*. Para hacer la explicación más comprensible, ejemplificamos cada patrón de forma separada. Nótese que la aplicación de los mismos permite, en cierta medida, resolver los problemas indicados en la sección 11.3

11.5.1 Landmarks

Un usuario dentro de *UniVirtual* generalmente interactúa con la aplicación en el contexto de diferentes secciones como son las noticias, su página personal o realizando búsquedas dentro del sitio.
Si bien estas secciones pueden tener información muy poco relacionada, permitir que el usuario acceda ágilmente a ellas es muy necesario; para simplificar este proceso usamos *landmarks*.
En *UniVirtual* los landmarks aparecen para acceder a las tres secciones mencionadas anteriormente: buscar, mi *UniVirtual* y la sección de noticias como mostramos en la Figura 11.15.

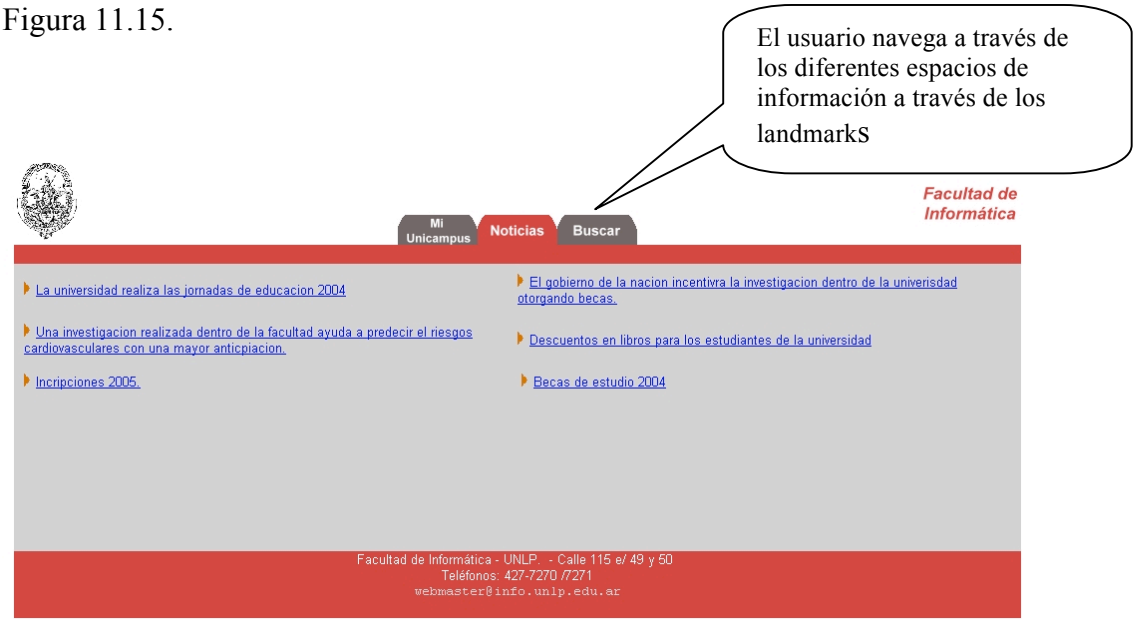


Figura 11.15: Landmarks en univirtual

11.5.2 Novedades

En *UniVirtual* la información está estructurada en forma de árbol. Si bien bajo esta estructura se tienen organizados correctamente los ítemes de información, los nuevos eventos que se producen pueden no ser descubiertos mediante navegación en profundidad en el árbol. Para evitar que los usuarios tengan que acceder a esta información de manera explícita, se aplica el patrón *Novedades* para mostrar los 12 eventos más próximos en el tiempo que ocurrirán en el campus. Debido a la gran cantidad de eventos que ocurren, se muestran cuatro de cada categoría.
Respetando la solución que propone el patrón, en la página inicial se muestra un enlace con un resumen del evento, que permite navegar a la información completa del mismo (Figura 11.16).

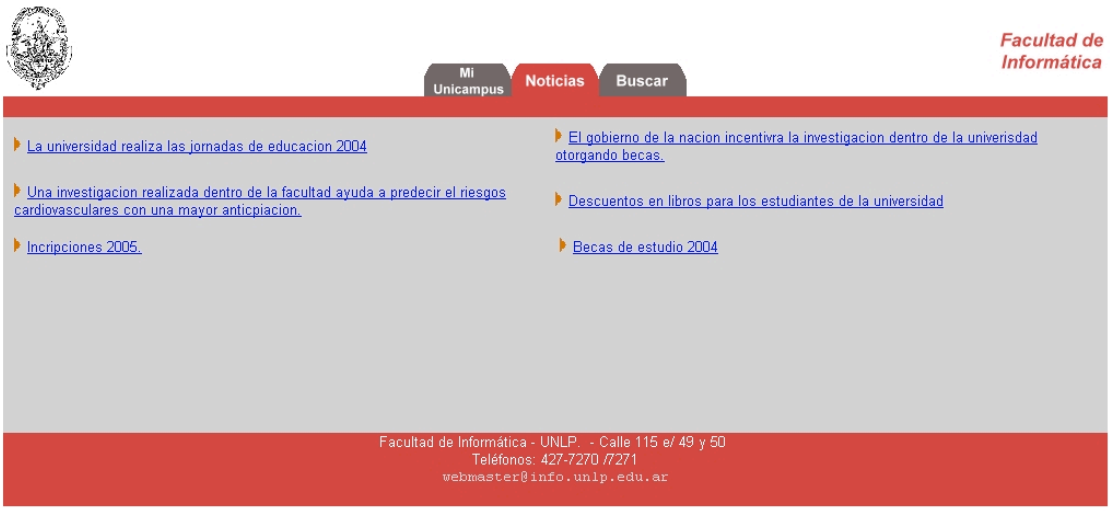


Figura 11.16: Novedades en *UniVirtual*

11.5.3 Comunicación Push

La solución previa es satisfactoria, sólo en caso que el usuario ingrese al sitio frecuentemente para ver las novedades y/o navegar. Sin embargo muchos usuarios pueden encontrar inconveniente este tipo de acceso, por ejemplo por falta de tiempo. Se propone entonces que tanto los eventos del campus (académicos y culturales) como los cursos extracurriculares tengan tanto acceso *pull* como *push*. Brindamos acceso *push* de dos maneras:
1-El usuario se suscribe brindando la dirección de correo electrónico en la cual quiere recibir la información.
2-El usuario se suscribe indicando su número de teléfono. De esta manera la información llega vía mensajes cortos (SMS o MMS).
En la Figura 11.17 se muestra parte del proceso de suscripción.

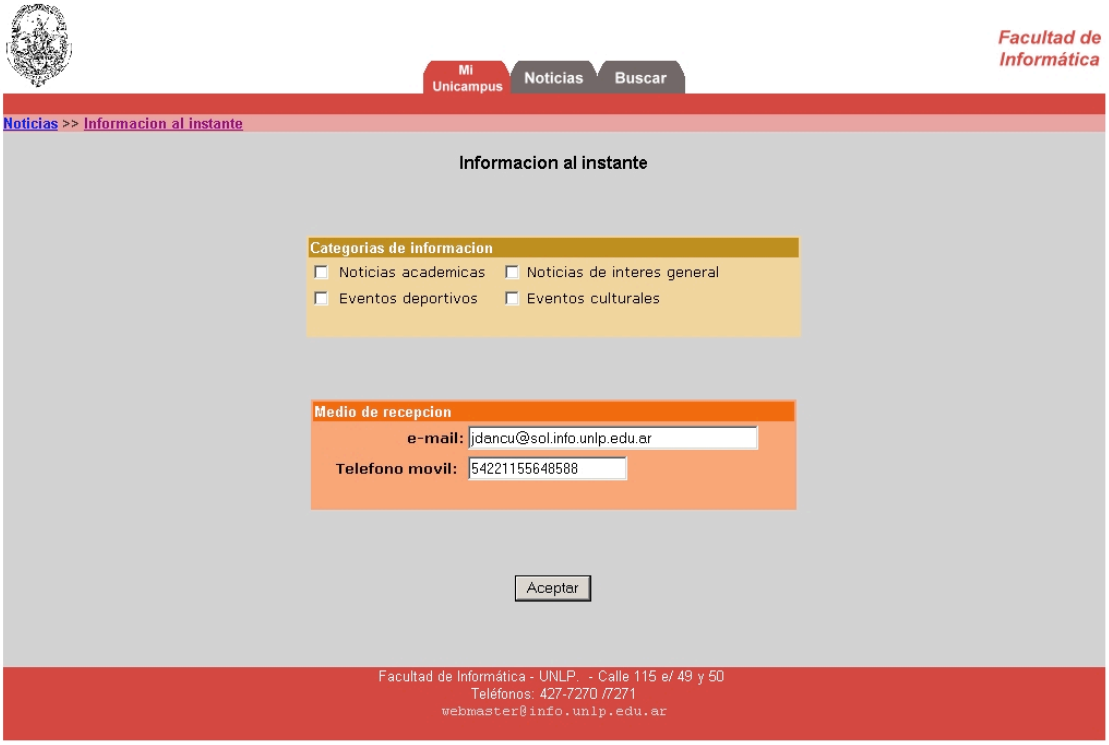


Figura 11.17: Comunicación Push en *UniVirtual*

11.5.4 Personalización de la estructura

El sistema de información de nuestra aplicación no sólo tiene un gran volumen, sino que también se caracteriza por contener información heterogénea y dinámica. Podemos encontrar desde información institucional de la Universidad hasta información personal del usuario como por ejemplo las calificaciones obtenidas en determinadas asignaturas. Como en otras aplicaciones de tipo portal, corremos el riesgo de provocar una sobrecarga cognitiva al mostrar demasiada información y además heterogénea.
Una buena solución a este problema consiste en agregar algún tipo de soporte de personalización. El usuario puede crear su propio *escritorio* dentro del campus virtual seleccionando las categorías de información que desea recibir. Por ejemplo: noticias y dentro de éstas que tipo desea ver (institucionales, deportivas, académicas), las n primeras citas de su agenda, un conjunto de sus enlaces favoritos, etc. Después de realizar esta configuración el usuario puede ingresar a su *escritorio web* a fin de tener disponible la información más relevante según su criterio como se muestra en la Figura 11.18.

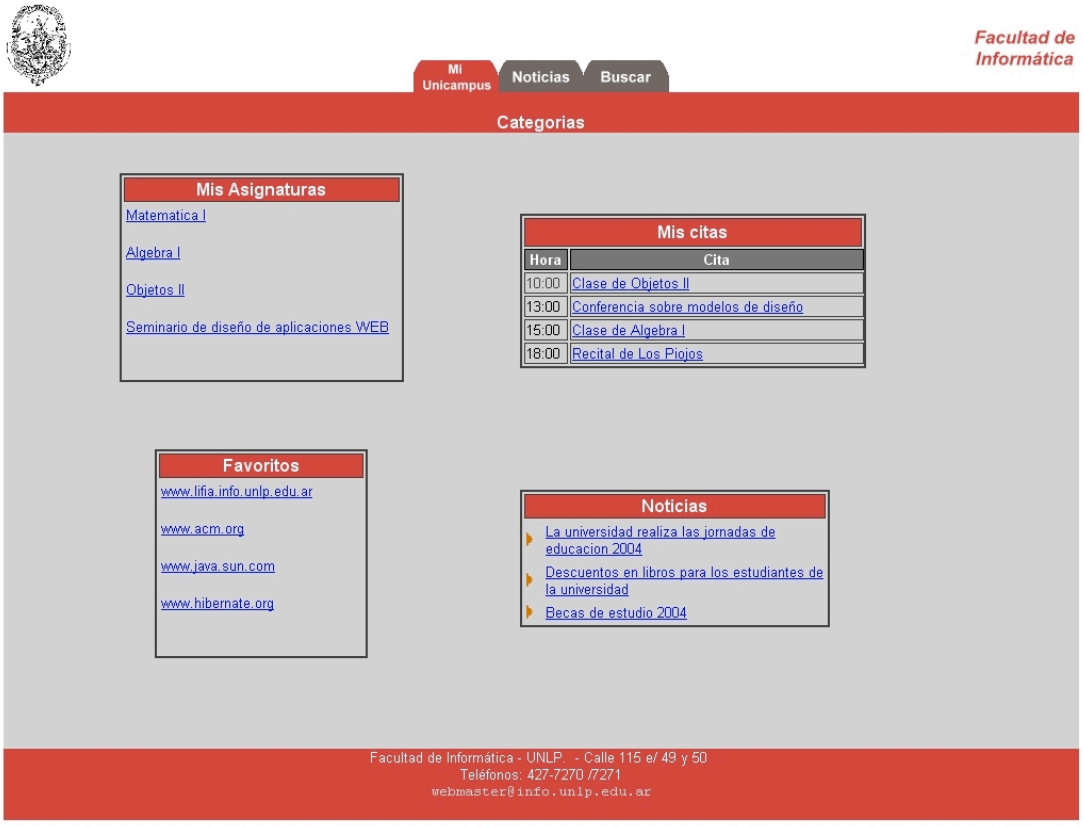


Figura 11.18: Personalización de estructura en *UniVirtual*

11.5.5 Espacio de Búsqueda:

Dentro de *UniVirtual* el usuario interactúa con un gran número de categorías de información y además de gran heterogeneidad. Las búsquedas globales, retornan muchos ítemes de información, algunos de los cuales pueden no ser de interés del usuario y la información buscada puede llegar a quedar "escondida" dentro de la

respuesta. Mediante la selección del espacio de búsqueda podemos resolver este problema.
Dada la complejidad del dominio se decidió utilizar un esquema de árbol de categorías en lugar de una estructura plana. De esta manera el usuario podrá navegar en el árbol hasta que encuentre la categoría deseada. Una vez seleccionada la categoría., la búsqueda se acotará a la categoría seleccionada y a sus descendientes como se muestra en la Figura 11.19.

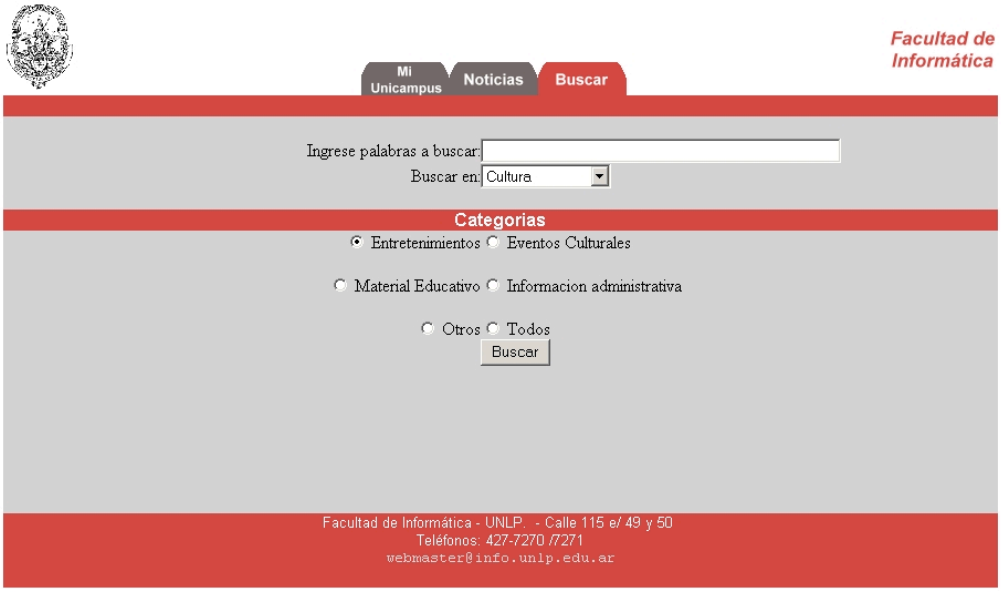


Figura 11.19: Espacios de búsqueda en *UniVirtual.*

11.5.6 Respuesta Estructurada.

Dentro de *UniVirtual* la gran cantidad de información y su diversidad de fuentes puede complicar la presentación del resultado de una búsqueda.
Supongamos que un usuario realiza una búsqueda sobre la palabra clave "automóvil", el resultado de las búsquedas puede retornar desde un enlace a un comercio que tiene convenio con la universidad para el alquiler de automóviles hasta el enlace a una tesis de grado sobre la polución ambiental producida por los automóviles.
A fin de brindarle al usuario una manera amigable para determinar la calidad de las respuestas, los resultados de las búsquedas son categorizados de acuerdo con las características de la información encontrada como se ilustra en la Figura 11.20.

Figura 11.20: Respuesta Estructurada en *UniVirtual*

11.5.7 Enlace Tipado

Dentro de *UniVirtual* la información presentada al usuario puede aparecer en varios formatos. En algunos casos los usuarios descargan material de una determinada materia que está disponible como archivos .zip, .pdf, .doc, etc.
Para evitar la confusión del usuario y facilitarle el proceso de selección del formato en que desean que la información les sea presentada se les ofrece una serie de gráficos representativos de dichos enlaces como mostramos en la Figura 11.21.

Figura 11.21: Link tipado en *UniVirtual*

11.5.8 Información bajo demanda

En muchos casos nos enfrentamos con la necesidad de no saturar al usuario con información y al mismo tiempo aprovechar el mismo espacio de interfaz para presentar diferentes datos, como por ejemplo para los investigadores de la Facultad. En las Figuras 11.22 y 11.23 mostramos de qué manera utilizamos el patrón Información bajo Demanda para resolver el problema.

Figura 11.22: Uso de información bajo demanda en *UniVirtual*

Figura 11.23: Información complementaria bajo demanda en *UniVirtual*

11.6 Conclusiones

En este capítulo hemos mostrado de qué manera podemos utilizar patrones web en el diseño de nuevas aplicaciones. Hemos ilustrado nuestro punto de vista con un conjunto de patrones descubiertos tanto en nuestros propios desarrollos como en los de otros sitios web exitosos. Nuestra experiencia con la metodología OOHDM nos ha demostrado que un conocimiento adecuado de un conjunto importante de patrones permite complementar el soporte conceptual dado por una metodología, cuando deben resolverse problemas complejos. Mediante el uso de una aplicación

arquetípica, *UniVirtual,* ejemplificamos la aplicación de los patrones a problemas específicos del diseño.
A medida que las aplicaciones web comienzan a hacerse más complejas y a abarcar ciertos aspectos novedosos (v.g soporte de procesos de negocios, movilidad y ubicuidad, etc.), la necesidad de registrar, transmitir y reusar experiencia de diseño se hace mas evidente; consideramos que los patrones son un vehiculo excelente en esta dirección.
Actualmente estamos trabajando en varias direcciones: por un lado buscamos mejores medios para integrar patrones en el contexto de notaciones de diseño, como por ejemplo la provista por métodos del estilo de OOHDM u otros como WebML o UWE. Al mismo tiempo seguimos reflexionando sobre el diseño de aplicaciones exitosas para descubrir estructuras de diseño recurrente, en dominios tales como aplicaciones móviles y sensibles al contexto.

11.7 Referencias

[Batory 00] Batory, D.: Refinements and Separation of Concerns. Second Workshop on Multi-Dimensional Separation of Concerns, International Conference on Software Engineering, Ireland, 2000. In http://www.cs.utexas.edu/ftp/pub/predator/separation.pdf.

[Ceri 00] Ceri, S., Fraternali, P., Bongio, A. (2000). Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. Proc WWW9 Conference, Amsterdam, NL, May 2000 (also in Computer Networks, 33 (2000), pp. 137-157).

[Gamma 95] Gamma, R., Helm, R., Johnson, R., Vlissides, J. "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley, 1995

[Hennicker 00] Hennicker, R. , Koch, N. (2000). A UML-based Methodology for Hypermedia Design. In A. Evans, S. Stuart, and B. Selic, editors, UML'2000 - The Unified Modeling Language - Advancing the Standard, volume 1939 of Lecture Notes in Computer Science,York, England, October 2000. Springer Verlag.

[HPat 02] Hypermedia Pattern Repository. ACM/SIGWEB and University of Lugano, Switzerland. In http://www.designpattern.lu.unisi.ch/index.htm

[Lyardet 98] Lyardet, F. Rossi, G., Schwabe, D. Discovering and using patterns in the WWW (Multimedia Tools and Applications, Kluwer, 1998

[Lyardet 99] Lyardet, F. Rossi, G., Schwabe, D. Patterns for adding search capabilities to Web Information Systems". Proceedings of EuroPLoP'99, the European Conference on Pattern Languages of Programs, IEEE Press, 1999

[Rossi 99a] Rossi, G., Schwabe, D., Lyardet, F. Integrating Patterns into the Hypermedia Development Process. The New Review of Hypermedia and Multimedia, Taylor Graham, December, 1999.

[Rossi 99b] Rossi, G., Schwabe, D. Web application models are more than conceptual models, Lecture Notes in Computer Science 1727, pp. 239-252, ISBN 3-540-66653-2, Proceedings of the World Wild Web and Conceptual Modeling'99 Workshop, ER'99 Conference, Springer, Paris, 1999

[Rossi 00a] Rossi, G., Lyardet, F., Schwabe, D. User Interface Patterns for hypermedia applications. Proceedings of the ACM Conference on Advanced Visual Interfaces 2000, Sorrento, Italy, May, 2000.

[Rossi 00b] Rossi, G., Lyardet, F., Schwabe, D. Patterns for E-commerce applications. Proceedings of EuroPLoP'00. IEEE Press.

[Schwabe 98] Schwabe, D., Rossi, G. (1998).An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems 4(4), 1998. Wiley and Sons, New York, ISSN 1074-3224).

[Schwabe 99] Schwabe, D., Rossi, G., Lyardet, F. Improving Web Information Systems with navigational patterns" International Journal of Computer Networks and Applications, May, 1999.

12 Integración de patrones en el proceso de diseño de aplicaciones hipermedia

Susana Montero, Paloma Díaz e Ignacio Aedo

Laboratorio DEI. Departamento de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad 30. 28911 Leganés, España
smontero@inf.uc3m.es, pdp@inf.uc3m.es, aedo@ia.uc3m.es
http://www.dei.inf.uc3m.es

12.1 Introducción

En la primera parte de este libro se ha puesto de manifiesto la gran relevancia que han adquirido los sistemas hipermedia, y en particular las aplicaciones web, cuyas principales señales de identidad son: estructuras de navegación cada vez más sofisticadas, comportamientos interactivos más complejos, composiciones multimedia que sean estéticas, información personalizada tanto para usuarios como para dispositivos, y control de acceso a dicha información, así como la necesidad de hacer frente al desarrollo de este peculiar tipo de aplicaciones mediante métodos de ingeniería hipermedia en las que las necesidades del usuario potencial o final guían el proceso de desarrollo con el objetivo de construir un producto completo, correcto, útil y utilizable.
Sin embargo, dos cuestiones quedan pendientes. Por un lado, el tiempo de desarrollo e implantación de proyectos web no debe superar en la mayoría de los casos los tres meses [7] y, además, un diseñador, tanto si es experto como no, puede llegar a tomar una solución inadecuada a un determinado problema de diseño que puede ser perjudicial para el usuario (v.g. desorientación, sobrecarga de información, pobre usabilidad, insatisfacción y carencia de confianza en el servicio ofrecido), debido a que frecuentemente es un único diseñador el que acomete el desarrollo de estos sistemas y debe de hacer frente a requerimientos de muy diversa índole (v.g. técnicos, de seguridad, de interacción, etc.) para los cuales a veces puede no poseer las habilidades necesarias [9]. Por otro lado, si el desarrollo es llevado a cabo por un equipo de trabajo multidisciplinar, sus integrantes serán personas con diferentes habilidades y conocimientos, como puedan ser los autores de los contenidos, los diseñadores gráficos, los ingenieros del software, los programadores, los especialistas en comercialización, y, por supuesto, los usuarios finales, teniendo que trabajar todos ellos conjuntamente hacia la misma dirección: cumplir las expectativas de los usuarios [36].
Los patrones de diseño pueden constituir una solución a ambos problemas desde que estos documentan una solución a un problema recurrente encontrado durante el proceso de diseño, de tal manera que no sólo dejan constancia de esa experiencia de diseño, sino que posibilitan la reutilización de la misma experiencia una y otra vez en diferentes aplicaciones [5]. Con respecto a las cuestiones planteadas anteriormente, entre las bondades de los patrones de diseño, como ya se comentó en el capítulo 10, se encuentran la de facilitar el diseño, mejorar la documentación, ayudar a reestructurar siste-

mas existentes e incluso proporcionar un vocabulario común que permite la comunicación de las diferentes alternativas de diseño de manera eficiente [10], ayudando así, tanto al diseñador experto como al novel, a reconocer situaciones en las cuáles la reutilización del diseño podría o debería ocurrir en la medida en que los patrones capturan el conocimiento experto en la resolución de problemas software [8]. Además, el campo de la interacción persona-ordenador ha puesto de manifiesto que los patrones pueden ser utilizados como herramienta de comunicación en un diseño participativo entre usuarios y desarrolladores [13].
Por lo tanto, los patrones de diseño son ese mecanismo que puede ayudar a aliviar tanto el acometido de nuevos desarrollos como la dificultad en la toma de decisiones para la mejora de sistemas ya existentes, en tanto en cuanto puedan integrarse de forma eficiente en los métodos de diseño y sean accesibles de manera efectiva por los diferentes integrantes del equipo de desarrollo.
El resto del presente capítulo profundizará en cómo afrontar dicha integración desde el contexto hipermedia. Para un mayor conocimiento sobre el origen y los conceptos básicos de los patrones véase la sección 10.1 del presente libro. En primer lugar, la sección 2 presenta y analiza algunos de los obstáculos que son necesarios superar para incrementar la utilización y aceptación de los patrones de diseño hipermedia. La sección 3 define las herramientas necesarias para asistir y guiar a los diseñadores en la búsqueda y aplicación de los patrones durante el proceso de diseño mediante un catálogo y un lenguaje de patrones. La sección 4 presenta cómo los patrones pueden ser integrados en el proceso de desarrollo de los métodos de manera beneficiosa ayudando al diseñador a dar el paso entre el análisis de requisitos y su transformación a entidades de diseño en el modelado conceptual, gracias a las soluciones capturadas por los patrones. Dicho proceso es aplicado a un caso de estudio. Además, la sección 5 presenta una formalización semántica que asienta las bases para el desarrollo de repositorios, herramientas y métodos para la organización, recuperación y exploración de los patrones existente de manera automatizada. Finalmente, la sección 6 enumera una serie de conclusiones y propone nuevos aspectos de trabajo.

12.2 Características y necesidades de los patrones de diseño hipermedia

Los patrones de diseño recogen el conocimiento y la experiencia de múltiples expertos, producto de muchos esfuerzos en el diseño y codificación de sistemas, para conseguir una mayor reutilización y flexibilidad del software. Sin embargo, a la hora de utilizar los patrones para resolver un determinado problema conviene conocer que la solución ofrecida por un patrón no es la panacea. Dado un problema, éste podría ser resuelto de muchas formas diferentes en función de los requisitos del sistema que se deseen satisfacer. Por esta razón es muy importante conocer cuándo es aplicable un patrón y cuáles son las consecuencias de su utilización. Además, el contexto hipermedia infiere a este problema dos nuevas perspectivas:

- **El proceso de diseño**. Los patrones de diseño hipermedia están pensados para capturar soluciones a problemas de diseño derivados del modelado (navega-

ción, presentación, dominio de la aplicación, comportamientos interactivos, personalización y seguridad), utilizando términos propios del área hipermedia y de las interfaces de usuario, sin llegar a describir aspectos de ejecución software. En concreto, se propone su utilización como mecanismo complementario a los métodos de diseño [39]. Esto implica que los patrones deben estar organizados de algún modo que permita encauzar los esfuerzos del diseñador a la hora de encontrar el patrón o patrones que mejor se adapten a las necesidades del problema, a la vez que se produce su integración con los métodos.

- **Los usuarios**. Los patrones de diseño hipermedia están dirigidos a diseñadores hipermedia, englobándose en este grupo los expertos del dominio, los especialistas en diseño, los lingüistas, los autores y los artistas, lo cual puede dificultar el reconocimiento de las situaciones en las cuáles los patrones de diseño se pueden reutilizar y cómo implementarlos con estructuras concretas que se ajusten a ellos. Esto implica que los patrones tienen que ser accesibles a personas con diferentes habilidades y formación.

En 1996, en el seno del grupo de investigación del LIFIA liderado por Gustavo Rossi surgen los primeros trabajos sobre patrones de diseño aplicados al desarrollo de aplicaciones hipermedia [41], siendo muchos los trabajos que siguieron y que fueron recopilados en [24]. Una vez que el dominio cuenta con un número considerable de patrones es necesario disponer de **catálogos** completos que ayuden a una recuperación sencilla y útil. En esta línea, el primer intento de catálogo recogía dos categorías, denominadas sistemas hipermedia y aplicaciones hipermedia, ésta última, subdividida en dos únicos aspectos de diseño (navegación e interfaz) [22]. Posteriormente, se presentaron diferentes clasificaciones como las de [34;23] pero sin los correspondientes patrones catalogados. Finalmente, en [24] se hizo un intento de recopilar todos los patrones publicados hasta la fecha pero no se le puede conceder el estatus de catálogo al no estar clasificados atendiendo a algún criterio. Por lo tanto, nos encontramos ante la carencia de un catálogo de patrones común que trate de recoger los patrones existentes entre publicaciones y repositorios web, lo que sigue haciendo inmanejable el proceso de búsqueda y selección de un patrón según el problema de diseño que estemos tratando. Además, estos patrones se caracterizan por promover la reutilización de macro-estructuras de diseño dependientes del método o modelo de diseño del cual el autor del patrón es experto, lo que dificulta su accesibilidad tanto a otros diseñadores expertos, como a los diseñadores noveles e incluso puede provocar el rechazo del resto de personas que se engloban como diseñadores hipermedia.

No obstante, aunque el catálogo permite una organización de los patrones, realmente no dirige el problema de la selección del patrón más adecuado dado un problema de diseño. En primer lugar, el diseñador debe comprender el esquema de clasificación usado en el catálogo. En segundo lugar, debe buscar la parte del catálogo donde podrían encontrarse aquellos patrones que son aplicables a su problema. Finalmente, aunque los patrones del catálogo puedan indicar de forma explícita la aplicación de otros patrones, esta orientación es sólo a nivel de patrones y no a nivel del problema que se está intentando resolver. Sin embargo, **un lenguaje de patrones** pro-

porciona una estructura de conocimiento que refleja las interrelaciones naturales entre los patrones, organizándolos como un todo, facilitando una solución detallada a un problema de diseño de gran escala cuyo propósito es el de guiar e informar al diseñador según atraviesa las relaciones de uso desde los patrones más generales a los más específicos [11]. En esta dirección se encuentran los nuevos patrones específicos para aplicaciones web que son presentados con un lenguaje más coloquial cercano al usuario de la aplicación final, bien orientado a diferentes tipos de sitios web y especializándose en patrones para comercio electrónico [12] u orientados a la usabilidad de la aplicación [26]. Sin embargo, su aplicación está más orientada al desarrollo de prototipos de manera *ad-hoc*, al no incluir macro-estructuras de diseño y centrarse en la apariencia de la interfaz, además de presentar soluciones dependientes de la tecnología web y en particular del lenguaje HTML.

Como última característica a destacar con respecto a los patrones de diseño hipermedia es su utilización como mecanismo complementario a los métodos de diseño, bien para solventar el hueco existente entre el modelado conceptual y la implementación del sistema, ya que los patrones permiten hacer una correspondencia entre el problema y su solución [23], o bien, para enriquecer dichos métodos con primitivas de más alto nivel de abstracción [40], es decir la aplicación de un patrón, incluyendo los elementos de diseño afectados, podría ser representada en el método con una única notación. Sin embargo, ninguna de estas dos aproximaciones ha tenido mucha aceptación por parte de los métodos hipermedia [33], en concreto por la falta de un vocabulario común independiente de métodos y modelos que permitiese una representación no ambigua para cualquier método.

Por lo tanto, para incrementar la utilización y aceptación de los patrones por parte del equipo de desarrollo es necesario, por un lado, proporcionar una serie de mecanismos que asistan y guíen a los diseñadores o desarrolladores de aplicaciones hipermedia durante el proceso de diseño y se adapten a sus diferentes habilidades y conocimientos, a la vez que se define un marco conceptual de alto nivel independiente de métodos o modelos de diseño, que permita la utilización de estos mecanismos de manera integrada con los métodos, con objeto de mejorar tanto la calidad del producto, introduciendo el conocimiento experto, así como la del proceso, a través de un lenguaje común de diseño hipermedia que favorezca la comunicación entre miembros de un equipo multidisciplinar, y conseguir así la madurez mostrada en otras áreas, como las de la ingeniería del software y la interacción persona-ordenador. La primera de las cuestiones se abordará en la sección 12.3, Herramientas para el uso de patrones, mediante el desarrollo de un catálogo y un lenguaje de patrones. La segunda de las cuestiones se planteará en la sección 12.4, Un diseño dirigido por patrones: aproximación y caso de estudio, mediante la especificación de un procedimiento de integración.

12.3 Herramientas para el uso de patrones

En la actualidad, si un diseñador o desarrollador de aplicaciones hipermedia quiere resolver o comunicar un problema específico de diseño con la ayuda de patrones hipermedia, como punto de partida debería buscar en las múltiples publicaciones existentes [12;15;18;22;23;24;26;38;41] o en alguno de los repositorios web disponi-

bles [1;43] Entre todos los patrones publicados, aproximadamente 200, el primer obstáculo evidente es determinar (P1) qué patrón o patrones pueden aplicarse a un determinado problema. Hasta la fecha no existe un catálogo o unos criterios comunes que permitan organizar los patrones de tal manera que su búsqueda sea más acotada, por lo cual el éxito de esta actividad dependerá del conocimiento que tenga el diseñador sobre los recursos existentes. Además, existe la posibilidad de encontrar ciertos patrones que describen el mismo problema pero desde diferentes puntos de vista [24], dificultando más aún esta actividad a los diseñadores poco experimentados. Una vez encontrado el patrón, el siguiente paso es (P2) cómo adaptarlo al contexto donde quiere ser aplicado. En el caso de los patrones hipermedia, los diseñadores se encuentran con un problema adicional. Cada patrón está expresado en términos del método o modelo de diseño utilizado por su autor o por el rol que éste desempeña en el proceso de desarrollo (v.g. autor, diseñador, artista o ingeniero). Así, la transferencia de conocimiento entre el patrón y el diseñador se ve comprometida al estar obligado a reinterpretar y reescribir el patrón en términos del método que él mismo utiliza. Finalmente, la aplicación de un patrón puede sugerir una serie de patrones relacionados que no tienen por qué coincidir con el problema global del diseño, por lo tanto si el diseñador desea saber (P3) cuál es el siguiente patrón a aplicar, debe volver a repetir este procedimiento de manera artesanal, por lo que el éxito de esta empresa estará ligado a la experiencia e intuición del diseñador.

El procedimiento anteriormente descrito, evidencia que el uso de los patrones no es un proceso trivial. Para afrontar cada uno de estos problemas se propone:

- Un **conjunto de criterios** que organice los patrones acorde a los diferentes aspectos de diseño a los que se enfrenta un diseñador hipermedia, así como a los diferentes niveles de conocimiento y de habilidad que pueda poseer, ayudando a organizar el espacio de patrones existente y el futuro (P1).
- La selección de un conjunto de patrones para formar parte de un **catálogo** destinado a facilitar la búsqueda de patrones en la resolución de problemas concretos de diseño (P1), organizado según los criterios propuestos.
- La reescritura de los patrones seleccionados en términos generales del dominio hipermedia haciéndolos tanto independiente de cualquier método o modelo de diseño como accesibles a todos los diseñadores (P2).
- La elaboración de **un lenguaje de patrones** orientado a ser guía activa durante el desarrollo de una aplicación hipermedia, aconsejando al diseñador como acometer dicha empresa desde diferentes niveles de abstracción, a la vez que le propone otros problemas a considerar (P3).

12.3.1 Organizando el espacio: un catálogo de patrones

Se puede considerar que cada patrón representa una categoría en sí mismo, por el hecho de presentar un plan general que puede ser utilizado para resolver un problema específico, expresando una relación entre un cierto contexto, un problema y una solución. Sin embargo, a medida que crece el número de patrones disponible es necesario

un modo de organizarlos en un nivel abstracto de conocimiento que facilite su localización.

Los criterios que a continuación se presentan tienen como objetivo categorizar los patrones hipermedia de acuerdo con la naturaleza del problema que resuelven, **el modelado de las aplicaciones hipermedia**, y a los diferentes **niveles de conocimiento y habilidades** que poseen los diseñadores hipermedia y usuarios que participan en el proceso de desarrollo de dichas aplicaciones. Consecuentemente, se ha determinado que el esquema de categorización esté formado por los siguientes ejes:

- **Aspecto de diseño**. Este criterio viene a recoger el primero de los objetivos, hacer explícito el aspecto de diseño en el que se puede englobar el problema que plantea el patrón definiendo así las siguientes categorías:
    - **Navegación**. Los patrones incluidos en este aspecto deberán hacer frente a problemas relacionados con el acceso a la información y la navegación a través de ella, de tal manera que el usuario no encuentre problemas de desorientación que contribuyan a la frustración de no localizar la información deseada.
    - **Estructura**. Estos patrones deberán asistir al diseñador en la organización de los diferentes tipos de información que componen una aplicación hipermedia pero de manera independiente al dominio, proporcionando una arquitectura que haga la navegación más intuitiva al usuario.
    - **Presentación**. Estos patrones deberán dar guías de cómo organizar los componentes de la interfaz para que la presentación de la información se haga de manera efectiva.
    - **Interacción**. Dentro de esta categoría se incluirán aquellos patrones que resuelvan problemas relacionados con el modo en que el usuario y la aplicación interactúan.
    - **Personalización**. Estos patrones establecen cómo proporcionar diferentes vistas y accesos a la información según las necesidades individuales de cada usuario o de cada grupo de usuarios.
    - **Seguridad**. Los patrones aquí recogidos deberán tratar cómo definir restricciones de seguridad en la aplicación para que se preserve la confidencialidad, la integridad y la disponibilidad de la información.
- **Nivel de descripción**. Este criterio recoge el segundo de los objetivos del catálogo haciendo explícitos los diferentes niveles de conceptualización encontrados durante el estudio de los patrones hipermedia, definiendo así las siguientes categorías:
    - **Alto**. Patrones que proporcionan el conocimiento necesario para que los destinatarios de las aplicaciones puedan participar en el proceso de diseño de una manera más activa, utilizando para ello un vocabulario más coloquial e independiente de los métodos de diseño.
    - **Medio**. Patrones que proponen la reutilización de macro-estructuras que han sido reconocidas como buenas prácticas y que se encuentran cercanas al modelado conceptual, utilizando para ello un vocabulario dependiente de los métodos de diseño dirigidos a expertos diseñadores hipermedia.

- **Bajo**. Patrones que reutilizan entidades de diseño para situaciones concretas con una implicación directa en la implementación y que son fácilmente reconocibles por todos los diseñadores hipermedia.

Las categorías aquí presentadas que cubren los criterios propuestos son fruto del estudio de los patrones publicados hasta la fecha, y de los métodos de diseño hipermedia en los que van a ser aplicados, por lo que queda abierta la inclusión de nuevas categorías en un futuro, si fuera necesario recoger nuevas características.

Una vez definido los criterios de clasificación, se dará paso a la categorización de los patrones que conformarán el catálogo. De la extensa bibliografía consultada se han seleccionados aquellos que mejor permitan determinar el alcance de este catálogo e incremente su potencial utilidad. Por lo tanto se han excluido aquellos patrones que:

- no resuelvan problemas específicos de las aplicaciones hipermedia (v.g. el patrón `Nodo como una vista de navegación` [22] que define un nodo hipermedia asociado a un objeto o grupo de objetos);
- no sean independientes del dominio de la aplicación (v.g. el patrón `Producto Virtual` [24] determina la información que debería aparecer para un producto de una tienda electrónica);
- no de solución a un problema de diseño conceptual (v.g. el patrón `Método de creación de un nodo` [22] que describe cuando un nodo debe ser generado de manera estática o dinámica, dependiendo de si los datos son almacenados en una base de datos o no).

A continuación se presentan los patrones seleccionados para formar parte del catálogo, organizados en diferentes tablas para mayor legibilidad. Cada tabla se corresponde con cada una de las categorías del criterio `Nivel de descripción`, dentro de la cual los patrones han sido clasificados por el `Aspecto de diseño` que la exposición del problema del patrón afronta, incluyendo además una breve frase que resume su intención. Por motivos de identificación se ha incluido junto al nombre del patrón un identificador formado por las siglas de su categoría de nivel de descripción (`A`-Alto, `M`-Medio y `B`-Bajo), de su aspecto de diseño (`N`-Navegación, `E`-Estructura, `P`-Presentación, `Z`-Personalización y `S`-Seguridad), y un número según su orden de aparición.

Tabla 12.1. Patrones hipermedia con un nivel de descripción alto

Navegación	
`[AN1]Diferentes modos de navegación` [15]	El usuario debe poder moverse por la información de diferentes modos según su motivación o intención
Estructura	
`[AE1]Estructura centrada en el usuario` [15;43]	El usuario debe poder acceder a la información y a sus tareas de una manera intuitiva y sencilla
Presentación	

[AP1]Estética [15;26;35]	El usuario debe encontrar siempre las herramientas de navegación y los contenidos de manera consistente e inequívoca a través de toda la aplicación a la vez que son presentados de una manera estética
Interacción	
[AI1]Interacción	El usuario debe poder controlar y conocer el proceso de interacción con respecto a la información y a las tareas que proporciona la aplicación
Personalización	
[AZ1]Personalización	El usuario debe poder acceder a una aplicación adaptada a sus intereses y necesidades
Seguridad	
[AS1]Control de acceso	El usuario debe acceder de manera controlada al sistema y autorizado para realizar ciertas tareas

Tabla 12.2. Patrones hipermedia con un nivel de descripción medio

Navegación	
[MN1]Índice [23]	El usuario debe poder acceder a un determinado miembro de un grupo de nodos de manera directa
[MN2]Visita guiada [23]	El usuario debe poder acceder a un grupo de nodos relacionados de manera sencilla
[MN3]Contexto de navegación [22]	El usuario debe poder explorar de diferentes maneras un nodo según la tarea que esté realizando
Estructura	
[ME1]Organización jerárquica [15]	El usuario debe poder moverse con familiaridad por grandes cantidades de información
[ME2]Organización basada en tareas [15]	El usuario debe poder completar un conjunto de tareas relacionadas de una manera rápida y sencilla
Presentación	
[MP1]Separación información-interacción [22]	El usuario debe poder diferenciar entre contenidos y tipos de control (navegación y no-navegación)
[MP2]Unión información-interacción [22]	El usuario debe de reconocer qué controles están asociados con qué contenidos

[MP3]Agrupación de acciones [22]	El usuario debe de reconocer los tipos de control que están relacionados
[MP4]Presentación multimedia [12]	El usuario debe percibir los elementos multimedia como una composición estética
[MP5]Canales de sincronización [12]	El usuario debe percibir los elementos multimedia como una composición armónica
Interacción	
[MI1]Información por demanda [22]	El usuario debe poder controlar la información extra que recibe en su pantalla
[MI2]Feed-back [22]	El usuario debe ser informado del estado de la interacción de tal modo que sepa qué esperar
Personalización	
[MZ1]Roles [20]	Los usuarios deben estar organizados por tipos según sus similitudes
[MZ2]Personalización de la estructura [38]	El usuario debe acceder sólo a aquellos nodos y contenidos que le interesan
[MZ3]Personalización del contenido [38]	El usuario debe recibir los contenidos de manera personalizada
Seguridad	
[MS1]Autorización [18]	El usuario debe de tener diferentes tipos de acceso a los recursos
[MS2]Control del acceso basado en roles [18]	Cada tipo de usuario debe tener los derechos de acceso que le correspondan
[MS3]Seguridad multinivel [18]	Cada usuario debe acceder sólo a aquellos recursos para los cuales tenga permisos

Tabla 12.3. Patrones hipermedia con un nivel de descripción bajo

Navegación	
[BN1]Migas de pan [15]	El usuario debe poder saber dónde se encuentra con respecto a la jerarquía de información
[BN2]Mapa [43]	El usuario debe poder saber dónde se encuentra dentro de la estructura de información de la aplicación
[BN3]Acción de búsqueda [15]	El usuario debe poder encontrar un elemento o información específica de manera rápida

[BN4]En un salto a casa [43]	El usuario debe poder llegar al nodo de entrada fácilmente desde cualquier sitio
Estructura	
[BE1]Centro de la colección [23]	El usuario debe saber qué tipo de información o tareas recoge un conjunto de nodos para poder seleccionar uno de ellos
[BE2]Nodo como una unidad única [22]	El usuario debe percibir que cada nodo es una unidad de información autocontenida y tiene sentido para él
[BE3]Página principal [15;43]	El usuario debe ser consciente del objetivo de la aplicación y cómo puede moverse para realizar sus tareas
Presentación	
[BP1]Barra de navegación [15;26;43]	El usuario debe encontrar siempre visibles y de manera consistente las herramientas de ayuda a la navegación
[BP1]Barra de pie de Página [43]	El usuario debe ser consciente de las condiciones de uso de la aplicación
Interacción	
[BI1]Botón de acción [15;43]	El usuario debe ser consciente de la importación de la acción en relación a otras acciones que se puedan realizar
Personalización	
Seguridad	

Los catálogos sirven para indexar los patrones por algún tipo de criterio, permitiendo que la búsqueda de una solución a un problema concreto sea más sencilla. Así, cuando el usuario se enfrenta a un problema de diseño, por ejemplo de navegación, y desea saber qué patrones utilizar, puede seleccionar alguno de los catalogados con este aspecto de diseño, e incluso según su habilidad o conocimientos reducir el número de patrones a los clasificados en un determinado nivel de descripción y ver cuál de ellos se adapta mejor a la situación actual de su diseño. Además, esta clasificación permite evidenciar qué aspectos son mejor cubiertos o no por los patrones existentes y dónde debería ponerse mayor énfasis para el descubrimiento de nuevos patrones (véase celdas vacías en las tablas), pudiendo ser incluidos posteriormente al catálogo.
Una descripción más detalla del contenido de estos patrones puede verse en el sitio web dedicado a los patrones de diseño hipermedia, http://pdh.dei.inf.uc3m.es.

12.3.2 Organizando el conocimiento: un lenguaje de patrones

En el catálogo anteriormente definido, los patrones han sido presentados simplemente como una solución a un problema concreto de diseño y clasificados en grupos según el aspecto de diseño que trataban y el nivel de descripción utilizado para hacer llegar su conocimiento al usuario, con el único objetivo de reducir el espacio de búsqueda de dichos patrones. Este tipo de categorización puede ser de utilidad para poder llevar a cabo una navegación dirigida por objetivos de diseño a través del espacio de patrones, como pueda suceder en otros catálogos de patrones hipermedia [1;43]. Sin embargo, diseñar una aplicación hipermedia requiere de la combinación de varios patrones, los cuales deben estar organizados y presentados de manera cohesiva, es decir, mediante un lenguaje de patrones en el cual cada una de las soluciones y de los nuevos problemas que plantee cada patrón estén orientados a resolver un problema común de ámbito general, en este caso, el diseño de una aplicación hipermedia.
El objetivo de este lenguaje de patrones es hacer explícito el conocimiento subyacente que existe cuando se combinan los patrones presentados en la sección anterior a través de una serie de enlaces estructurales que permiten ir moviéndose desde los patrones más genéricos a los más concretos, desde los que crean estructuras a los que añaden detalles a esas estructuras, y sucesivamente desde los que añaden detalles a los detalles, ayudando así al usuario a seleccionar el conjunto de patrones que le permitan desarrollar el núcleo de la aplicación hipermedia ajustándose a sus necesidades.
Para que la motivación y el alcance de este lenguaje se hiciesen más explícitos, se ha optado por dividir los patrones en grupos conceptuales que comparten un objetivo común dentro del diseño de una aplicación hipermedia, elaborándose expresamente para dicho fin una serie de patrones. El patrón `[A]Aplicación hipermedia` es el encargado de explicar esta división y la motivación de cada uno de esos grupos de patrones. Posteriormente, de igual manera se describe cada grupo a través de un patrón de los considerados como nivel de descripción alto en la Tabla 1 (`[AE1]Estructura centrada en el usuario`, `[AN1]Diferentes modos de navegación`, `[AP1]Estética`, `[AI1]Interacción`, `[AZ1]Personalización` y `[AS1]Control de acceso`) que recoge la esencia del grupo. En este caso particular, el campo **`patrones relacionados`** recoge las relaciones con aquellos patrones de menor escala necesarios para llevar a cabo la solución propuesta por el patrón mostrando así cada grupo conceptual como un todo a la vez que sirve de sumario e índice al lenguaje.

[A]Aplicación hipermedia

Contexto: Si la aplicación ha desarrollar tiene como objetivo facilitar el acceso y la manipulación de la información, la cual es representada mediante contenidos multimedia, a la vez que proporciona comportamientos interactivos para que el usuario pueda llevar a cabo alguna tarea, estamos ante una aplicación hipermedia. Además, puede ser necesario presentar la información de manera personalizada tanto para usuarios como para dispositivos, a la vez que se preserva la seguridad de la información.

Problema: ¿Cómo afrontar el diseño de una aplicación hipermedia para que sea útil y fácil de usar por el usuario?
Discusión: La hipermedia es una tecnología que está siendo utilizada en diferentes dominios como puedan ser bibliotecas digitales, sistemas de aprendizaje o comercio electrónico. Cada dominio de aplicación se caracteriza por una serie de conceptos y de relaciones entre sí que deben ser identificados con el objetivo de proporcionar una **estructura organizativa** a la aplicación, si fuese necesario, y un mayor conocimiento del dominio que representan [14].
La estructura no lineal de la hipermedia hace de la **navegación** uno de los puntos más importantes, ya que es mediante la selección de enlaces como el usuario puede moverse libremente por el espacio de información. Como consecuencia de ello, el usuario corre el riesgo de "perderse en el hiperespacio", alcanzando una posición de la cual es incapaz de salir hacia un punto conocido.
Por otro lado, la multimedia ha permitido integrar diferentes medios, tanto en cuanto, los nodos no incluyen sólo información textual, sino otros tipos de información como imágenes, sonido o vídeo. Esta inclusión en el hipertexto supone nuevos retos relacionados con la **presentación** de la información, es decir, con la necesidad de organizar y armonizar dicha información en diferentes dimensiones como el tiempo y el espacio bi- o tri-dimensional [34].
La comunicación entre el usuario y las aplicaciones hipermedia ha adquirido un alcance y extensión muy diferente al propósito para el cual fueron concebidas, ya no sólo proporcionan como funcionalidad principal la navegación a través de espacios de información sino que incorporan a su interfaz nuevos **comportamientos** interactivos, fruto de los cambios tecnológicos, además de la absorción de las funcionalidades de las aplicaciones software ya existentes, proporcionando nodos con contenidos interactivos, realidad virtual, facilidades de colaboración y comunicación, acceso a otros sistemas, etc. [14].
En esta nueva generación de aplicaciones hipermedia se está potenciando la accesibilidad mediante la **personalización**, tanto de la información como de las funcionalidades a diferentes audiencias y dispositivos. No todos los usuarios poseen las mismas motivaciones, conocimientos o preferencias por lo que es necesario adaptar el contenido de la aplicación y su utilización a las particularidades de los usuarios. La proliferación de dispositivos móviles, hace necesario no sólo tener en cuenta las preferencias de los usuarios sino también las de los entornos [27].
Esta última característica pone de manifiesto la necesidad de preservar la seguridad de la información contenida en las aplicaciones, mediante el uso de políticas de acceso que preserven la confidencialidad, la integridad y la disponibilidad de dicha información [2].
Todas estas características están intrínsecamente ligadas con el fin de dar lugar a sistemas útiles y fáciles de utilizar.
Solución: Abordar el diseño de la aplicación desde diferentes vertientes siguiendo una aproximación centrada en el usuario. Para ello es necesario determinar previamente las metas de la aplicación en cuestiones como la audiencia, el contenido, la funcionalidad o el aspecto. Posteriormente, organizar la información del dominio independientemente del sistema de navegación, teniendo en cuenta la perspectiva del usuario y cómo realiza éste sus tareas. Proporcionar diferentes esquemas de navegación que permitan al usuario encontrar lo que él quiere. Organizar los elementos que

aparecen en la interfaz de manera consistente e intuitiva a través de toda la aplicación, presentándolos de una manera atractiva. Proporcionar mecanismos que permitan al usuario controlar y conocer en todo momento el estado de la interacción con la aplicación. Además, si fuese necesario proporcionar diferentes vistas de la aplicación según las necesidades de los diferentes tipos de usuario. Controlar quién puede acceder a qué cosas dentro del sistema.

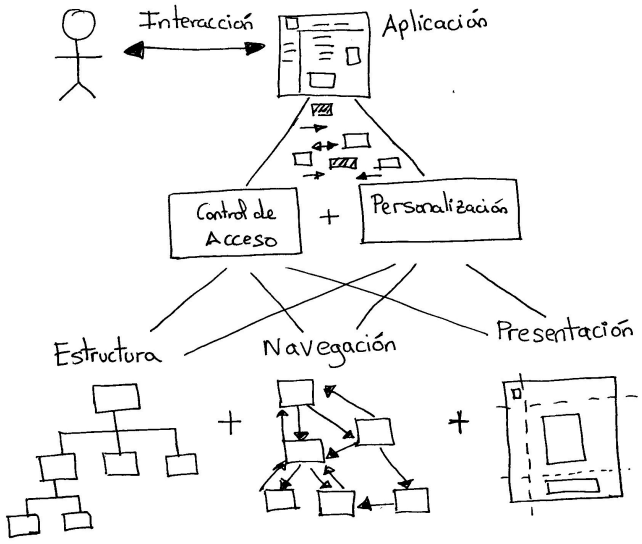


Figura 12.1. Separar la estructura de la información, de la navegación y de la presentación, hace que la aplicación sea accesible a un mayor número de personas y dispositivos

Patrones relacionados: El patrón `[AE1]Estructura centrada en el usuario` ayuda a organizar la información teniendo en cuenta las necesidades de los usuarios. El patrón `[AN1]Diferentes modos de navegación` guía al usuario tanto a la hora de explorar el espacio de información, como acometer sus tareas o encontrar lo que busca de manera rápida. En cuanto al diseño de la interfaz, el patrón `[AP1]Estética` acomete la combinación correcta de los elementos, tanto en su número como en sus relaciones espaciales y temporales, para que dichos elementos sean presentados de una manera efectiva. Además de la navegación es necesario mejorar la comunicación entre el usuario y la aplicación mediante `[AI1]Interacción`. Si se desea que la aplicación sea utilizada por el mayor número de usuarios, se pude seguir el patrón `[AZ1]Personalización`, a la vez que es necesario controlar quién puede hacer qué cosas en la aplicación, mediante el patrón `[AS1]Control de acceso`.

[AE1]Estructura centrada en el usuario

Contexto: Se está construyendo una `[A]Aplicación hipermedia` y se desea reflejar la estructura de su dominio.
Problema: ¿Cómo estructurar la información y las tareas de la aplicación para proporcionar un acceso intuitivo y sencillo al usuario?
Motivación: Las aplicaciones hipermedia cada vez tienen que albergar más y más cantidades de información, a la vez que deben proporcionar un acceso a la misma de una manera intuitiva y sencilla. Independientemente del diseño de la navegación que se elija para la aplicación, existe un espacio de información subyacente que debe ser organizado. Cuando se hace frente a un nuevo y complejo sistema de información, las personas desarrollan ciertas expectaciones de cómo encontrar los diferentes tipos de información y cómo llevar a cabo ciertas tareas en la aplicación, por ejemplo, cuando una usuario accede a una aplicación que ofrece servicios, espera que se le faciliten los servicios o información relevante sobre ellos, y no cómo está organizada la empresa de manera interna. El éxito de la aplicación estará determinado en gran parte por cómo de bien la organización de la información encaje con las expectativas de los usuarios.
Solución: Utilizar a algunos usuarios representativos que ayuden a organizar la información de la aplicación de tal modo que les parezca lo más lógica posible. Establecer el dominio de los usuarios identificando los conceptos, las tareas y las relaciones estructurales de la información que maximicen la accesibilidad y la manipulación de la información. Organizar la información en jerarquías de categorías que ayuden al usuario a encontrarlas. Organizar las operaciones según cómo les gustaría a los usuarios realizarlas.

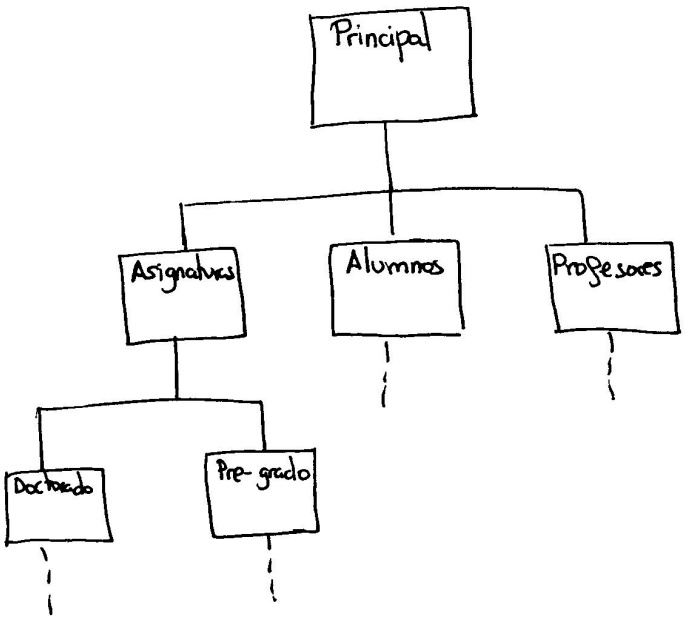


Figura 12.2. Balancear entre anchura y profundidad; si una jerarquía es demasiado ancha, los usuarios pueden tener que elegir entre demasiadas opciones; si es demasiado profunda, van a tener que navegar demasiado para alcanzar un determinado nodo

Patrones relacionados: Para organizar la información de manera jerárquica, utilizar `[AE2] Organización jerárquica`. Cuando exista una secuencia en la información para llevar a cabo tareas se debería utilizar una `[AE3] Organización basada en tareas`. Cada subcategoría de la estructura o tarea debería ser presentada al usuario con la ayuda de un `[BE1]Centro de colección`. Además, para definir lo que sería cada una de las unidades de información de la estructura, se puede consultar un `[BE2]Nodo como una unidad única`. Para finalizar se puede utilizar como nivel más alto de la estructura organizativa una `[BE3]Página principal`.

[AN1]Diferentes modos de navegación

Contexto: Se está construyendo una `[A]Aplicación hipermedia` y la información ha sido organizada siguiendo una `[AE1]Estructura centrada en el usuario`. Ahora, es necesario dotar al usuario de mecanismos de navegación y de acceso al espacio de información.
Problema: ¿Cómo hacer que la navegación sea clara y consistente a la vez que facilita encontrar la información deseada?
Motivación: En una aplicación hipermedia la interacción básica consiste en que el usuario seleccione los enlaces para moverse por un amplio espacio de información provisto de cientos de nodos y realizar tareas de diferentes modos. Este espacio es muy grande, y la navegación difícil, por tanto, es necesario ofrecer al usuario un soporte de navegación en el que pueda orientarse. Pero además, los usuarios suelen tener diferentes intenciones cuando navegan por la aplicación, por ejemplo pueden tener algo en mente y quieren acceder directamente a lo que buscan, otros prefieren navegar por una determinada parte de la aplicación siguiendo los enlaces, y otro tercer tipo de usuarios vagamente saben lo que quieren y desean poder echar un vistazo y ver que les interesa. El esquema de navegación seleccionado además debería proporcionar siempre respuestas claras y no ambiguas a dos cuestiones básicas: ¿A dónde puedo ir? y ¿Dónde estoy?[35]:

- **`Dónde puedo ir`**. Una ventaja importante es tener una buena estructura de información a la hora de poder ayudar al usuario a moverse por el espacio de información; y, sobre todo, a orientarle hacia dónde puede ir. Sin embargo, mientras los usuarios navegan a través de esa estructura, bien en profundidad o explorando diferentes enlaces, pueden llegar a sentirse perdidos.
- **`Dónde estoy`**. Quizá sea la referencia más importante de la navegación, ya que cualquier usuario no podrá entender la estructura de la aplicación si no sabe dónde está, y tampoco tendrá la capacidad de interpretar el enlace de dónde viene y a dónde puede ir. Por ello, es necesario resaltar la ubicación relativa con respecto a la estructura de la aplicación y mostrar el área dónde se encuentra un nodo concreto.

Solución: Ofrecer diferentes modos de navegar por la información para que se adapte a los diferentes tipos de usuarios. Ayudar a los usuarios a manejarse a través de gran-

des cantidades de información y a que completen sus tareas. Añadir herramientas de navegación hacia páginas relacionadas y superiores del espacio de información. Añadir herramientas de búsqueda que ayuden a encontrar rápidamente la información deseada.

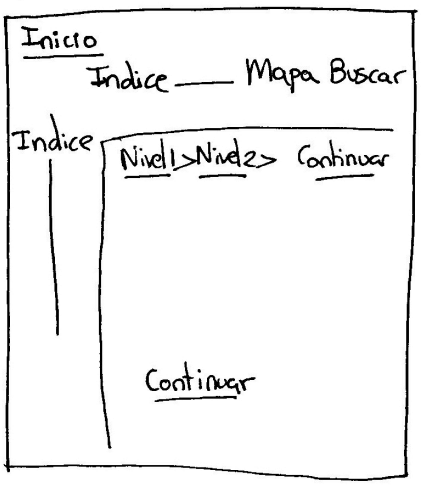


Figura 12.3. Proporcionar a los usuarios diferentes modos de navegar por el sistema para que utilicen la que mejor se adapte a sus metas y deseos

Patrones relacionados: El patrón `[MN1]Índice` ayuda a los usuarios a seleccionar un destino, mientras que el patrón `[MN2]Visita guiada` ayuda al usuario a completar sus tareas de manera guiada. Se puede hacer que el espacio de navegación sea explorado por diferentes criterios o contextos con `[MN3]Contexto de navegación`. Para orientar al usuario sobre su estado de navegación actual con respecto a una rama del espacio de información se aconseja utilizar el patrón `[BN1]Migas de pan`. Para mostrar la situación del usuario con respecto al espacio de información total de la aplicación, utilizar el patrón `[BN2]Mapa`. Para que el usuario pueda conseguir la información deseada de forma directa, se puede proporcionar una `[BN3]Acción de búsqueda`. Los usuarios deberían ser capaces de volver a la página principal o a otros puntos mayores de navegación en un sólo paso con `[BN4]En un salto a casa`.

`[AP1]Estética`

Contexto: Se está diseñando una `[A]Aplicación hipermedia` y la información ha sido organizada mediante unidades lógicas según una `[AE1]Estructura centrada en el usuario` a la vez que se han proporcionado `[AN1]Diferentes modos de navegación`. Ahora la apariencia de cada una de esas unidades de información debe ser definida.

Problema: ¿Cómo organizar la interfaz de la aplicación para que sea consistente e intuitiva a la vez que los contenidos son mostrados de manera estética?
Motivación: La consistencia y la predecibilidad son atributos esenciales de cualquier sistema de información bien diseñado. Una aplicación hipermedia que no es consistente de nodo a nodo hace difícil la navegación por parte de los usuarios. Una aproximación consistente de presentación y navegación permite que los usuarios se adapten rápidamente al diseño y facilita la predicción de la localización de la información y de los controles de navegación de la aplicación. Los esquemas de diseño que subyacen bajo la mayoría de las publicaciones de papel bien maquetadas son igualmente de necesarios a la hora de diseñar documentos electrónicos y publicaciones *online*, en las que las relaciones espaciales entre los elementos de la pantalla están constantemente variando en respuesta a las peticiones del usuario y la actividad del sistema. Además, los usuarios pueden sentirse confundidos si esperan encontrar el nombre de la sección en un determinado lugar y no la encuentran.
Solución: Establecer un esquema y estilo para presentar el texto y las imágenes, aplicándolo de forma consistente para crear unidad entre los diferentes nodos de la aplicación. Mantener todos los elementos de contenido y navegación en los mismos lugares en cada nodo de forma que el usuario reconozca la estructura y sienta que todavía se encuentra en el lugar adecuado.

Figura 12.4. Repetir no es aburrido, hay que proporcionar una identidad consistente que refuerza la sensación de "unidad" y haga que el usuario reconozca la aplicación

Patrones relacionados: Para que los accesos a la información estén siempre visibles y consistentes, utilizar el patrón `[BP1]Barra de navegación`. Para ayudar al usuario a diferenciar entre contenidos y tipos de control (navegación y no- navegación) consultar el patrón `[MP1] Separación información-interacción`, a la vez que reconoce qué controles están asociados con qué contenidos con los patrones `[MP2]Unión información-interacción` y `[MP3]Agrupación de comportamiento`. Para mostrar los elementos multimedia de una manera armónica

utilizar el patrón [MP4]Presentación multimedia y [MP5]Sincronizar canales. Si la aplicación tiene condiciones de uso legales deberían colocarse siguiendo el patrón [BP2] Barra de pie de página.

[AI1]Interacción

Contexto: Se está diseñando una [A]Aplicación hipermedia y uno de los modos de interacción es la navegación definida en [AN1]Diferentes modos de navegación, pero es necesario mejorar la comunicación entre el usuario y la aplicación.
Problema: ¿Cómo hacer para que el usuario sienta que controla la interacción?
Motivación: La interacción implica un diálogo entre dos partes. En el contexto de la hipermedia, la interactividad es la funcionalidad proporcionada por un sistema para responder [16]. Al principio las aplicaciones hipermedia eran sólo utilizadas para proporcionar información. Sin embargo, el uso de la Web ha hecho que se expanda hacia un uso más general como servicios de noticias, servicios de banca y compra *online*, en los cuales los usuarios además de navegar a través del espacio de información pueden realizar una serie de tareas que provocan cambios de estado en el sistema, a veces irreversibles. Cuando el usuario utiliza un sistema interactivo necesita tener la impresión de que en todo momento es él el que controla la marcha de los procesos, sino se impacientará y cometerá errores, como ejecutar la misma opción repetidamente.
Solución: Dotar al usuario de más control sobre sus acciones. Hacer distinguibles aquellos elementos cuya activación provocan navegación de aquellos que permiten realizar tareas. Informar al usuario del efecto que provocan sus acciones en el sistema.

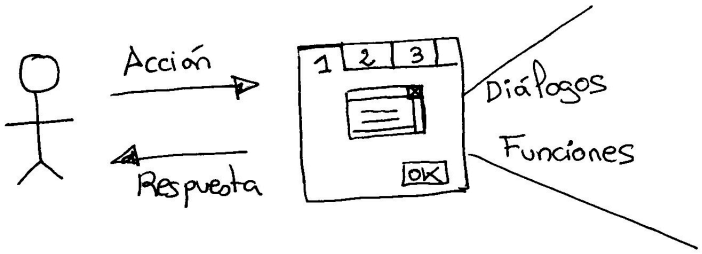


Figura 12.5. Informar al usuario sobre el efecto de sus acciones permitiéndole evaluar su objetivo final a alcanzar y determinar su acción futura para conseguirlo

Patrones relacionados: Para que el usuario sea el que controle cuánta información desea visualizar en un nodo utilizar el patrón [MI1]Información por demanda. Para informar al usuario del estado de la interacción de tal modo que sepa qué esperar consultar el patrón [MI2]Feed-back. Aquellos elementos cuya activación provoquen la ejecución de una acción deberían ser presentados como [BI1]Botones de acción.

[AZ1]Personalización

Contexto: Se está diseñando una [A]Aplicación hipermedia donde se ha definido una [AE1]Estructura centrada en el usuario, diferentes tipos de acceso con [AN1]Diferentes modos de navegar, el aspecto estético y cognitivo de los contenidos con [AP1]Estética y cómo responder a la interacción con el usuario en [AI1]Interacción, todo ello separando cada uno de los aspectos de diseño. Ahora se desea considerar adaptar la aplicación a las necesidades de cada usuario.
Problema: ¿Cómo proporcionar una aplicación personalizada sin renunciar a la modularidad y al fácil mantenimiento?
Motivación: La gran cantidad de información residente en las aplicaciones hipermedia hace de la navegación de los usuarios una tarea dura. La personalización hipermedia es el proceso de adaptar la información o servicios proporcionados por la aplicación a las necesidades de cada usuario específico o conjunto de usuarios, a partir de los intereses individuales, en combinación con el contenido y la estructura de la aplicación. Debido a que la personalización es un aspecto crítico en muchos dominios como el comercio electrónico, es importante tratarlo como una vista de diseño, más que como una vista de implementación. Por lo tanto, no se trata de construir aspectos específicos de estas aplicaciones, sino reutilizar aquellos aspectos que son comunes a la mayoría de ellos [32;39].
Solución: Analizar los diferentes tipos de usuarios. Agruparlos por características comunes. Ofrecer diferentes vistas del mismo espacio de información a cada tipo de usuario a través de un conjunto de reglas de presentación.

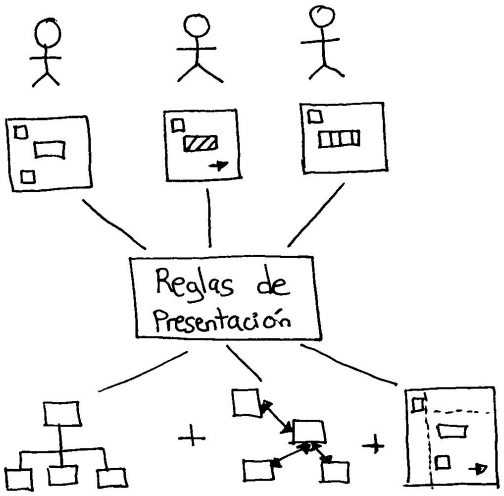


Figura 12.6. Separar la estructura de la información de los contenidos para que cada usuario obtenga una vista diferente sin que el mantenimiento de la aplicación se vea penalizado

Patrones relacionados: Para definir los diferentes tipos de usuarios de la aplicación utilizar [MZ1]Roles. Si se quiere restringir el espacio de visualización a los intereses de cada tipo de usuario consultar [MZ2]Personalización de la es-

tructura. En el caso de querer personalizar los valores de los contenidos usar [MZ3]Personalización del contenido.

[AS1]Control de acceso

Contexto: Se está diseñando una [A]Aplicación hipermedia donde existen diferentes tipos de usuarios que pueden estar recogidos mediante [MZ1]Roles. Tanto la información como los servicios definidos en la [AE1]Estructura centrada en el usuario deberían tener un acceso controlado.
Problema: ¿Cómo determinar políticas de seguridad sin afectar a la estructura y navegación ya definidas?
Motivación: El control de acceso es un requerimiento esencial en los sistemas hipermedia y en especial en los basados en web. La mayoría de las compañías y organizaciones no sólo tienen disponible información a través de sitios web sino que también proporcionan servicios más avanzados a usuarios autorizados como acceso a información sensible, transacciones financieras, etc. No se trata por tanto, de personalizar los contenidos que son mostrados al usuario, sino mantener el control sobre la información personal. Por lo tanto es necesario definir las limitaciones de los usuarios en términos de lo que pueden hacer, a qué recursos pueden acceder y qué funciones se les permite realizar sobre los datos. Durante el proceso de desarrollo de los sistemas web e hipermedia, los requerimientos de acceso, tienen que ser abordados desde diferentes niveles de abstracción [19] y no sólo como decisiones de implementación.
Solución: Cuantificar el valor relativo de la información a proteger en términos de Confidencialidad, Sensibilidad, Clasificación, Privacidad e Integridad, relacionándolo tanto con la organización como con los usuarios individuales. Determinar la interacción relativa que los propietarios y creadores de los datos tendrán dentro de la aplicación web. Algunas aplicaciones pueden restringir el acceso a todos los usuarios excepto a los autores de los datos y a los administradores del sistema. Además hay que determinar qué funciones específicas de usuario deberían ser construidas dentro de la aplicación web (v.g. identificación de usuarios, ver su información, modificarla, etc.) así como las administrativas (cambiar contraseñas, ver los datos de algún usuario, etc.) [6].

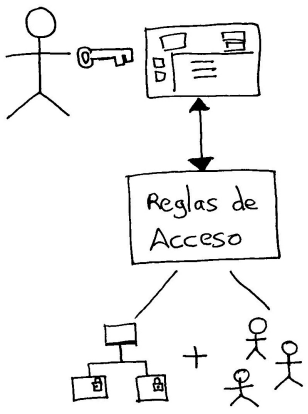


Figura 12.7. Controlar el acceso a los usuarios para que accedan sólo a aquellos recursos para los cuales tengan permiso

Patrones relacionados: El control de acceso a los recursos es descrito en `[MS1]Autorización`. Una especialización de este patrón se encuentra en `[MS2]Control de acceso basado en roles`. Además se pueden asignar diferentes niveles de manipulación a la información con `[MS3]Seguridad multinivel`.

En la siguiente sección, se describirá el proceso que permite aplicar este lenguaje de patrones en combinación con los métodos de diseño hipermedia.

12.4 Un diseño dirigido por patrones: aproximación y caso de estudio

Los métodos de diseño hipermedia, como los descritos en la Parte II, Ingeniería de la web, de este libro, tienen la cualidad de guiar al diseñador en su tarea de modelar diferentes aspectos de las aplicaciones hipermedia como la navegación, la estructura del dominio de la aplicación, la presentación, el comportamiento, la personalización e incluso la seguridad a través de un conjunto de actividades y fases que permiten aproximar dichos aspectos de una manera sistemática e iterativa hacia el producto final. Sin embargo, esto puede llevar a los diseñadores a redescubrir soluciones a problemas de diseño comunes sin beneficiarse de cómo fueron resueltos en el pasado, dando como resultado un esfuerzo duplicado, un diseño inconsistente y sistemas frágiles y de pobre trazabilidad. Una aproximación más efectiva es construir un sistema hipermedia a partir de patrones bien documentados. El catálogo de patrones presentado en la sección 12.3.1 documenta las mejores prácticas de diseño hipermedia, capturando las propiedades esenciales de usabilidad comunes a los mejores ejemplos de diseño. Por lo tanto un patrón de diseño hipermedia puede verse como **`una regla que transforma el sistema para alcanzar ciertos requisitos de usabilidad`**. Además, en la sección 12.3.2 se presentó un lenguaje de patrones cuyo objetivo es guiar al diseñador durante el proceso de diseño mediante una red de patrones relacionados, en el cual cada uno contribuye a resolver un problema común, el diseño de una aplicación hipermedia.

Teniendo en cuenta estos dos elementos, el catálogo y el lenguaje, a continuación se presenta **`un proceso de diseño guiado por patrones`** que complementa a los métodos de diseño, ayudando a los diseñadores y desarrolladores hipermedia a incluir el conocimiento experto en sus propios diseños. Los patrones de diseño hipermedia siempre se han propuesto como una técnica complementaria a los métodos, para solventar el hueco existente entre la especificación de requisitos y el modelado conceptual [23].

En el proceso de desarrollo de una aplicación hipermedia, como aparece en la Figura 12.8 adaptado de [28], durante la fase de análisis de la aplicación se identifican las necesidades del usuario definiendo lo que sería el **`espacio del problema`**, las cuales son documentadas como **`requisitos`**. Posteriormente, es en la fase de dise-

ño donde se describe **el espacio de la solución** para esos requisitos. Por lo tanto, el diseñador necesita ser capaz de sintetizar un problema para producir una solución completa y eficiente.
Es en este punto donde los patrones pueden jugar un papel relevante, reduciendo el paso entre estos dos estados. En concreto, debido al formato en que son presentados los patrones, como una regla que guía la transformación entre un problema dado y una solución, esto permitiría hacer una **correspondencia entre problemas conceptuales y soluciones de modelado**. Este razonamiento puede verse reflejado en la Figura 12.8 en el hecho de que los requisitos de la aplicación son identificados con la parte del problema del patrón (P), y cómo la solución (S) determina un modelo de diseño concreto.

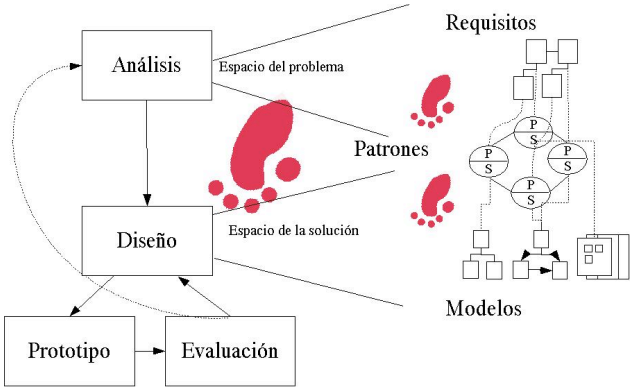


Figura 12.8. Integración de los patrones en el proceso de desarrollo de las aplicaciones hipermedia

A continuación se enumeran los pasos a seguir para que los requisitos guíen una aproximación basada en patrones para afrontar el modelado conceptual:

- **Paso 1.** Obtención de los requisitos software usando alguna de las técnicas propuestas en [17].
- **Paso 2.** Asignar a cada requisito funcional aquel patrón o patrones que mejor se adapten a él. Para ello el diseñador se ayudará del catálogo de patrones recogido en las tablas presentadas en la subsección 12.3.1, las cuales incluyen un resumen de la intención de cada patrón. Los patrones seleccionados deberían de recaer en los niveles de descripción `Medio y Bajo`, ya que son dichos patrones los que capturan soluciones aplicables a modelos conceptuales.
    - **Paso 2.1.** Si alguno de los patrones seleccionados cae en la categoría de nivel de descripción `Alto` se puede deber a dos motivos: el requisito es demasiado general y es necesario volver al Paso 1, en este caso el patrón seleccionado puede ayudar a concretar esos nuevos requisitos; o ha habido una equivocación en la selección y es necesario repetir el Paso 2.
- **Paso 3.** Refinar los patrones seleccionados con ayuda del lenguaje de patrones, el cual fue presentado en la subsección 12.3.2. A partir del campo `patrones relacionados` seleccionar aquellos patrones de menor escala que soportan a ese patrón, e incluir aquellos que puedan completar al requisito.

- **Paso 3.1.** Si alguno de los nuevos patrones seleccionados cae en la categoría de nivel de descripción `Alto` repetir el Paso 3 hasta que todos los niveles de descripción de los patrones sea `Medio y Bajo`.

- **Paso 4**.Organizar los requisitos de acuerdo con el aspecto de diseño al que pertenezca el patrón (estructura, navegación, presentación, interacción, personalización, seguridad) y determinar las prioridades entre ellos. Este paso sirve de preparación para el último paso a la vez que permitirá una entrada ordenada a los productos de los métodos.
- **Paso 5.** Adaptar la solución del patrón al dominio de la aplicación. Es necesario identificar aquellos elementos que participan en la solución y los pasos a seguir para aplicarla.
- **Paso 6.** Instanciar la solución con los correspondientes elementos de diseño del método. Cada elemento identificado en el paso anterior deber corresponderse con algún elemento del diseño conceptual, generando una primera aproximación al modelado de la aplicación.

Este conjunto de pasos puede ser visto como un proceso iterativo ya que puede ser necesario refinar los requisitos a través de los nuevos problemas planteados por los patrones seleccionados en los pasos 2 y 3.
Finalmente, mencionar que este proceso no pretende ser ortogonal al uso común de un lenguaje de patrones, sino que sea utilizado de manera complementaria cuando se enfrente el diseñador ante un nuevo desarrollo de una aplicación hipermedia a partir de un conjunto de requisitos identificados.

12.4.1 Un caso de estudio: ARCE

Para ilustrar el proceso descrito en el apartado anterior, se utilizará un subconjunto de los requisitos del sistema ARCE, ya presentado en la sección 7.6 de este libro, los cuales recogen sus características funcionales más relevantes. En general, el sistema proporciona mecanismos para notificar una emergencia, pedir recursos y ofrecer asistencia. Cuando una emergencia ocurre, el país afectado, utiliza ARCE para informar a la asociación sobre la situación. Cuando se requiere asistencia, los recursos son clasificados según un glosario multilingüe incluido en el sistema, la petición de ayuda es dada de alta, los asociados son notificados por correo electrónico para que puedan acceder al sistema y ver qué recursos son necesarios y cómo pueden ayudar. Para cada emergencia, la aplicación proporciona información actualizada sobre qué recursos son necesarios, las cantidades iniciales y cuántos elementos han sido ya facilitados, permitiendo así que cada asociado decida cómo contribuir [3]. La Tabla 4 lista en su columna izquierda un subconjunto de los requisitos implicados en el diseño del sistema ARCE, y en su columna derecha los patrones que ayudan a resolver los problemas conceptuales planteados por dichos requisitos. Ambas columnas son el resultado de los pasos 1, 2 y 3 del proceso mencionando anteriormente en esta sección, `obtener requisitos`, `seleccionar patrones` y `refinarlos`.

Tabla 12.4. Requisitos para el sistema ARCE

R1. Modos de operar:	Paso 2:

Normalidad: R1.1 Los usuarios autorizados pueden publicar noticias que podrán ser leídas por cualquiera de los usuarios del sistema. R1.2. Los usuarios autorizados pueden intercambiar mensajes entre ellos. Emergencia: R1.3 El país afectado creará un informe inicial de la situación de emergencia. R1.4 Otros países socios podrán ofertar medios movilizables. R1.5 El país afectado podrá realizar una solicitud detallada de sus necesidades de manera cuantificada. R1.6 Los usuarios autorizados podrán consultar la situación de las aportaciones formuladas por otros países.	[ME1]Organización jerárquica Aspecto de diseño: **Estructural** Paso 3: [BE2]Nodo como una unidad única
R.2 Todos los datos serán manejados a través de formularios que son posteriormente procesados.	Paso 2: [MP2]Separación información-interacción Aspecto de diseño: **Presentación**
R.3 Existirá un histórico con todas las situaciones de emergencias cerradas.	Paso 2: [BE2]Nodo como una unidad única Aspecto de diseño: **Estructural**
R.4 Tanto las operaciones cotidianas como las de emergencia deberán estar siempre accesibles.	Paso 2: [MN1]Índice [MP1]Barra de Navegación Aspecto de diseño: **Navegación**
R.5 Un glosario unificado de términos relacionados con la defensa y la protección civil deberá estar siempre accesible con el objetivo de unificar la terminología sobre emergencias.	[MP1]Barra de Navegación Paso 2: [BE2] Nodo como una unidad única Aspecto de diseño: **Estructural**
R.6 Todos los procesos, tanto los de normalidad como de emergencia, deberán ser activados por usuarios autorizados. Se	Paso 2: [MS2]Control de acceso basado en roles

establecen los siguientes tipos de usuarios: **Nacionales**: Pertenecientes a un país. **Organismos asociados**: Direcciones Generales de Defensa / Protección Civil que incluyan el Centro Coordinador Operativo H24 (sólo puede existir uno por cada país). **Organismos no asociados**: Organismos y organizaciones invitadas, al sistema, para visualizar información (Organismos, Organizaciones, Instituciones públicas, ONGs). **Supranacionales**: Asociaciones que incluyen a varios países. **Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil**. **Internacionales**: Organizaciones que no pertenezcan a un único país y que puedan visualizar informaciones (ONU, ...)	Paso3: `[MZ1]Roles` Aspecto de diseño: **`Seguridad`**
R.7 La diversidad de países requiere un soporte multilingüe al menos para dos lenguas de la comunidad Latino-Americana que tienen que ser soportadas tanto por el interfaz de usuario como por el glosario de términos.	Paso 2: `[MZ3]Personalización de contenidos` Aspecto de diseño: **`Personalización`**

Los tres últimos pasos que restan del procedimiento serán explicados sobre una parte concreta del ejemplo, como son los requisitos que fueron cubiertos por los patrones estructurales. Del paso 4 se obtiene que esos requisitos son R1, R3 y R5. Todos ellos tienen en común que recogen las necesidades del usuario en cuanto a las operaciones e información que espera poder realizar y consultar en el sistema, y son cubiertos por el patrón `[ME1]Organización jerárquica` y `[BE2]Nodo como una unidad única`. El último paso de este proceso tiene como objetivo adaptar la solución del patrón al dominio en el que se está trabajando e instanciarlo en entidades concretas de diseño. En la Figura 12.9 puede verse el resultado total de aplicar este paso a cada uno de los requisitos. El patrón `[ME1]Organización jerárquica` expone en su solución seguir tres pasos:

1. Dividir el espacio de información en nodos. Realmente en la descripción textual del patrón se plantea como un nuevo problema que es resuelto por otro patrón de menor nivel `[BE2]Nodo como una unidad única`, en el cual se define que `un nodo debería incluir una unidad de información autocontenida que tenga sentido para un conjunto de usuarios`. Aplicándolo al dominio de la aplicación, de los requisitos se obtienen las siguientes unidades de información o nodos, que en algunos casos representan las tareas que desean realizar los usuarios: **`Publicar noticias`**, **`Intercam-`**

biar mensajes, **Informe inicial**, **Solicitud detallada**, **Listado de aportaciones**, **Histórico de emergencias** y **Glosario**.

2. Organizar los nodos por importancia y grado entre ellos. De los requisitos se obtienen que hay tareas que están agrupadas bajo dos categorías diferentes **Normalidad** y **Emergencia**, y todas ellas tienen igual importancia para el usuario.
3. Establecer las relaciones estructurales entre los nodos. En este punto, en la Figura 12.9 puede verse el resultado de instanciar este patrón utilizando los elementos de diseño de un método, en concreto Ariadne el cual fue introducido en el capítulo 2. Este método utiliza como relaciones estructurales la generalización y la agrupación, que permiten describir que el sistema ARCE está formado de operaciones de Normalidad y de Emergencias (R1), un Histórico (R3) de emergencias y un Glosario (R5).

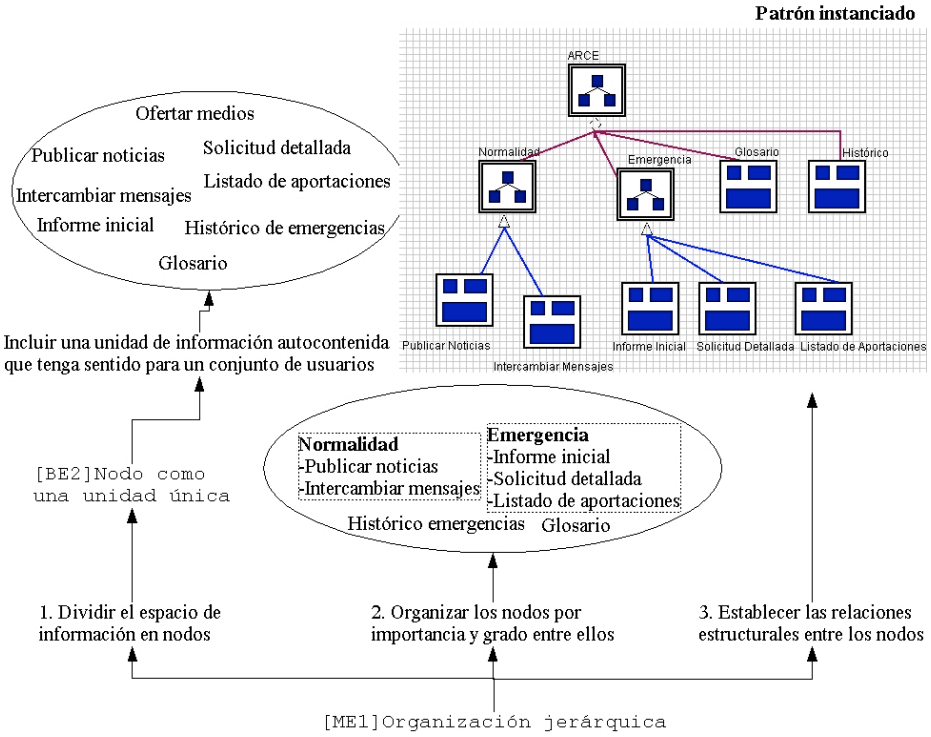


Figura 12.9. Instanciación del patrón [ME1]Organización jerárquica para el sistema ARCE

Para finalizar esta sección, señalar que el resultado de este proceso es una primera aproximación guiada al modelado conceptual del sistema, el cual deberá ser refinado en sucesiones iterativas incluyendo tanto la propia experiencia del diseñador como aquellas modificaciones fruto de las evaluaciones de utilidad y usabilidad realizadas sobre los prototipos.

12.5 El siguiente paso: una especificación formal

La gran cantidad de patrones de diseño hipermedia existentes se encuentran repartidos entre publicaciones [12;15;18;22;23;24;26;38;41] y algunos sitios web [1;43] en los que se incluye una mera copia de la presentación tradicional sin ninguna otra funcionalidad, que les pueda aportar un valor añadido a este formato. Es por esto que se necesita desarrollar repositorios, herramientas y métodos para buscar, seleccionar o explorar los patrones existentes sin tener que recorrer cada uno de estos recursos de manera individual, y extender el empleo de patrones en el proceso de desarrollo software. Además, un patrón de diseño sistemáticamente nombra, propone, explica y evalúa una solución para un problema de diseño recurrente de manera narrativa, aunque es indudablemente interesante desde un punto de vista pedagógico, puede plantear problemas de ambigüedad o precisión, sobre todo si se pretende acometer, por ejemplo, su aplicación de manera automatizada. Para ayudar a los usuarios a hacer frente a las dificultades de entender cuándo y cómo usar los patrones, actualmente se están proponiendo especificaciones más formales que aporten tanto una descripción semántica bien definida como un modelo de razonamiento, con el objetivo de complementar la descripción textual con la que son descritos los patrones y así permitir el desarrollo de herramientas software para incorporar sistemáticamente patrones en el diseño de sistemas [29], verificar su presencia [4], o encontrar aquellos que sean más adecuados a un determinado problema [25].

Hasta la fecha se puede afirmar que el campo de la Ingeniería de la Web no cuenta con una aproximación para la formalización de patrones, y de las propuestas realizadas en el dominio de la ingeniería del software encontramos que adoptan una semántica matemática para describir el dominio de la orientación a objetos que no es legible para personas sin conocimientos teóricos en ambos dominios, lo que frena su utilización por parte de diseñadores noveles o de un equipo de desarrollo multidisciplinar, como es el caso del desarrollo web. Además, sólo la solución del patrón es formalizada dejando de lado su descripción textual, tan importante para entender el fundamento que subyace en el problema y la solución que plantean. Todos los elementos del patrón son necesarios para desarrollar herramientas efectivas para su organización, recuperación y exploración, como por ejemplo incluir la intención de patrón permitiría indexar los patrones según el problema que resuelven, ayudando a los usuarios a encontrar el patrón correcto para una solución de diseño [30]. Sin embargo, todavía deben añadirse dos características más, propias de los patrones de diseño hipermedia que obstaculizan más aún una aplicación directa de los fundamentos teóricos de formalización del dominio de la ingeniería del software. Por un lado, la heterogeneidad tanto en su forma como en su contenido, en contraposición a los patrones contenidos en [21] objeto de todas las aproximaciones de formalización existentes. En el dominio web encontramos que cada autor utiliza su propia plantilla para describir los patrones de diseño hipermedia, existiendo una gama que va desde el patrón reducido a su mínima expresión (problema-solución), al que combina diferentes campos en el mismo (v.g. el problema y el contexto del patrón), lo que dificulta la comprensión e intencionalidad del mismo para usuarios noveles. Además, a pesar de que los patrones hipermedia describen los problemas de un dominio concreto, no hay un consenso a la hora

de describir dichos patrones, encontrándonos ante dos estilos narrativos. El primero engloba los patrones recopilados hasta [24], con un lenguaje mucho más técnico en el sentido de que son dependientes de alguno de los modelos o métodos de diseño hipermedia, lo que dificulta su uso tanto a los diseñadores expertos, que deben adaptar el contenido del patrón a los elementos de diseño del método que estén utilizando, como a los diseñadores noveles que más que dificultad lo que les puede provocar es rechazo. El segundo estilo narrativo, representado por [15;26] consiste en un relato mediante un lenguaje más coloquial cercano al usuario de la aplicación final.
Todas estas cuestiones conllevan a problemas de ambigüedad semántica lo que dificulta el desarrollo de herramientas de soporte común. Por lo tanto, en el caso de los patrones de diseño hipermedia es necesario contar con un mecanismo formal que permita esta pluralidad de formas y contenidos, a la vez que respete la descripción textual del patrón, medio de comunicación de su conocimiento entre los diseñadores.
En [31] se ha propuesto una representación semántica basada en ontologías que combina tanto el conocimiento del dominio hipermedia utilizado para la descripción de los patrones como el formato del patrón, definidos ambos de manera independiente. Posteriormente, esta representación es utilizada como armazón subyacente para complementar la representación textual del patrón mediante anotaciones semánticas [30]. Esta combinación de representación formal enlazada a su representación textual mediante anotaciones hace a los patrones mutuamente comprensible a diseñadores y a programas.

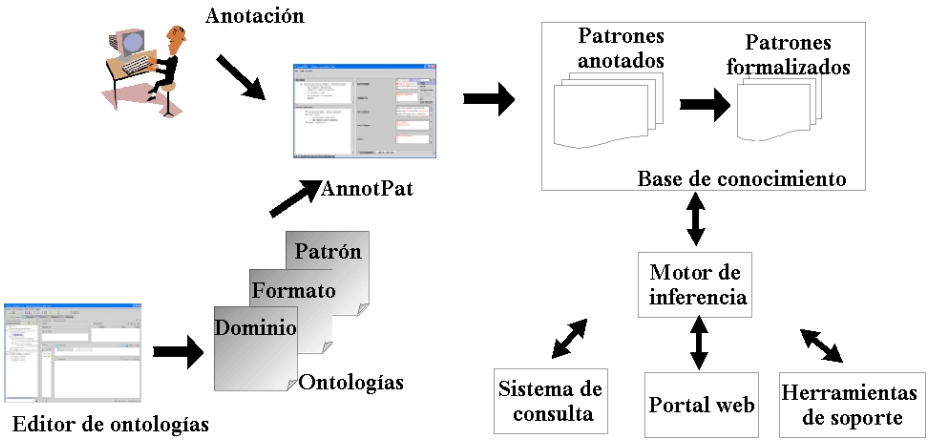


Figura 12.10. Proceso de anotación para la formalización de los patrones de diseño

La Figura 12.10 muestra el proceso de anotación que podría ser realizado por el autor del patrón o un diseñador hipermedia. Para soportar dicho proceso se ha desarrollado una herramienta de anotación [30], llamada AnnotPat, que proporciona como funcionalidades principales anotar la descripción textual de un patrón con los términos semánticos apropiados. Las ontologías que proporcionan la representación semántica de un patrón deben ser proporcionadas a la herramienta con el objetivo de que pueda guiar al usuario durante el proceso de anotación. De tal manera, después de dicho proceso, se extrae el patrón formalizado y se almacena junto a su descripción textual,

para su posterior procesamiento. A partir de la base de conocimiento que forman los patrones junto con un motor de inferencia, diferentes herramientas pueden ser desarrolladas, como herramientas de consulta para buscar y recuperar patrones apropiados en términos de la ontología, un portal web a través del cual el conocimiento de los patrones pueda ser recopilado, almacenado, y accedido por los miembros de la comunidad, o herramientas de soporte que interactúen con otros sistemas que ayuden al diseñador a seleccionar y aplicar patrones según el contexto de la aplicación. Todos los lenguajes utilizados tanto para especificar las ontologías como los patrones formalizados y las anotaciones son los propuestos por el W3C (*World Wide Web Consortium*) para estas materias, OWL (*Web Ontology Language*) [42] y RDF (*Resource Description Framework*) respectivamente [37].

La decisión de utilizar las ontologías como mecanismo de representación se debe más allá de la moda que padecen en este momento. Las ontologías han demostrado ser instrumentos de representación y organización del conocimiento proporcionando los siguientes beneficios a la propuesta de formalización aquí presentada para los patrones de diseño hipermedia:

- **Semántica bien definida**. Permiten conocer de forma precisa el sentido de los términos y conceptos utilizados en la descripción del patrón.
- **Razonamiento riguroso**. Proporcionan servicios de razonamiento sobre el patrón formalizado, puesto que los lenguajes de especificación de ontologías existentes están normalmente basados en un tipo particular de lógica formal, permitiendo a los programas software interpretar los patrones.
- **Búsqueda/Navegación inteligente**. Posibilitan que los repositorios de patrones sean gestionados por motores de búsqueda o agentes que den soporte a usuarios no sólo para la recuperación de patrones sino también para su descubrimiento.
- **Interoperabilidad**. Posibilitan que diferentes repositorios con patrones formalizados de manera diferentes coexistan, es decir, intercambien sus patrones de diseño, además de ser capaces de reutilizarlos.

Además, el uso de ontologías como formalismo subyacente para realizar las anotaciones sobre texto ordinario, permite la formalización de todos los campos del patrón, manteniendo su esencia intacta, a la vez que la formalización se mantiene transparente al usuario y sólo es utilizada para el tratamiento computacional, como puede verse en la Figura 12.11, donde la parte izquierda representa la vista literaria del patrón `[MN1]Índice,` y la parte derecha su formalización.

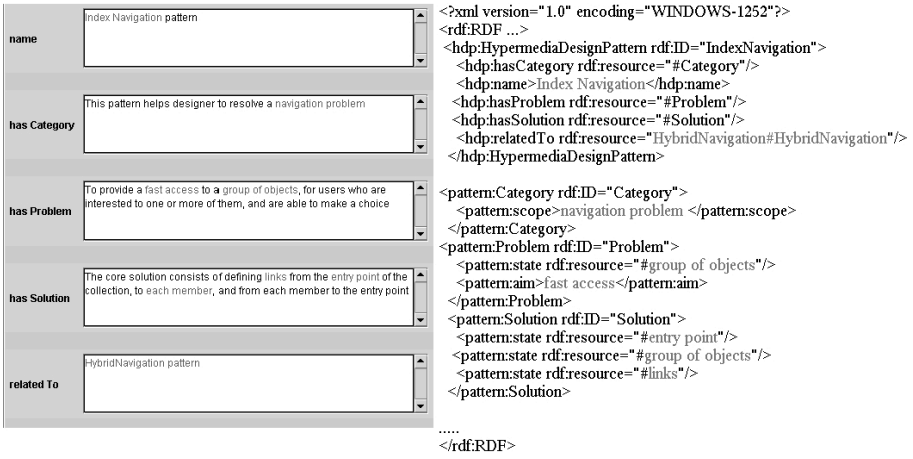


Figura 12.11. Las dos vistas del patrón `[MN1]Índice`

12.6 Conclusiones

En este capítulo se han analizado cuáles son los principales obstáculos que dificultan la utilización de los patrones de diseño dentro del proceso de desarrollo de las aplicaciones hipermedia, y de su poca aceptación por parte de los diseñadores hipermedia, a pesar de sus probados beneficios. De este estudio surge la necesidad de elaborar una serie de herramientas, como son un catálogo, un lenguaje y un proceso de integración con los métodos diseño, cuyo objetivo común es asistir y guiar a los diseñadores en la búsqueda y aplicación de los patrones durante el proceso de diseño.

Una vez que el espacio y el conocimiento de los patrones están organizados, los esfuerzos deberían ir dirigidos hacia la especificación de una formalización que permita su tratamiento por programas software. En esa dirección, se ha presentado una aproximación basada en ontologías que permite a los diseñadores enriquecer sus patrones de diseño con anotaciones semánticas usando los conceptos y términos propios del dominio hipermedia, proporcionando así el fundamento formal para construir repositorios de patrones estandarizados y herramientas que permitan automatizar su organización, recuperación y exploración. La principal ventaja de esta aproximación es su dualidad, siendo legible por los diseñadores ya que el texto del patrón está unido a su representación formal, y protegiendo así la esencia del patrón del diseño “expresar y comunicar soluciones exitosas a problemas comunes entre diseñadores”[21].

Muchos esfuerzos son todavía necesarios para una mayor aceptación y difusión de los patrones. En términos generales, un punto importante es la evaluación de su utilidad y usabilidad por parte de diseñadores noveles y expertos, a partir de la cual poder mejorar aspectos como su formato, narrativa o enfoque. Otra dirección a mejorar, sería fomentar la faceta de los patrones como mecanismo de comunicación entre los miembros del equipo de desarrollo de las aplicaciones hipermedia, en el que se incluye a los usuarios finales o potenciales del sistema, sobre todo como ayuda en la captura de

requisitos o como herramienta educativa para la transmisión de los conocimientos sobre el modelado hipermedia a través de ejemplos.
En cuanto al trabajo aquí presentado, hay que poner especial atención al lenguaje de patrones, ya que surge de su uso cotidiano, y al igual que el lenguaje natural, va evolucionando. Esto significa que continuamente van a emerger nuevas categorías de patrones y nuevos patrones que deberían ser tomados en cuenta. El proceso que lleva acabo la integración de los patrones en los métodos de diseño puede ser mejorado incluyendo el efecto Delta introducido en el capítulo 10, con objeto de restringir y guiar la selección de patrones entre las diferentes iteraciones del proceso. Finalmente, una acción inmediata es la implementación de dicho proceso en una herramienta software como pueda ser AriadneTool, presentada en el capítulo 2.

Agradecimientos

Este trabajo está financiado por la Dirección General de Investigación de la Comunidad Autónoma de Madrid y el Fondo Social Europeo con la referencia 07T/0024/2003-1. ARCE es un proyecto que se ha realizado en cooperación con la Dirección General de Protección Civil y Emergencias en el marco de los correspondientes convenios específicos.

12.7 Referencias

[1]ACM-SIGWEB y the University of Italian Switzerland. The Hypermedia Design Patterns Repository http://www.designpattern.lu.unisi.ch (2002)
[2]Aedo, I., Díaz, P. y Montero, S. A methodological approach for hypermedia security modeling. Information & Software Technology, 45(5): (2003) 229–239
[3]Aedo et al, 2002. Supporting Efficient Multinational Disaster Response through a Web-Based System. Proceedings of First International Conference of Electronic Government. Aix-en-Provence, France, (2000) 215-222
[4]Albin-Amiot, H. y Guéhéneuc, Y. Meta-modeling design patterns: Application to pattern detection and code synthesis. In Proceedings of the ECOOP Workshop on Automating Object-Oriented Software Development Methods (2001) 57–64
[5]Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M,. Fiksdahl-King, I. y Angel, S. A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
[6]Access Control and Authorization. A Guide to Building Secure Web Applications http://www.cgisecurity.com/owasp/html/ch08.html (2002)
[7]Barry, C. y Lang, M. A survey of multimedia and web development techniques and methodology usage. IEEE Multimedia (2001) 52–60
[8]Beck, K., Crocker, R., Meszaros, G., Vlissides, J., Coplien, J. O., Dominick, L. y Paulisch, F. Industrial experience with design patterns. In Proceedings of the18th international conference on Software engineering. IEEE Computer Society (1996) 103–114
[9]Bolchini, D. Web design patterns: Improving quality and performance in web application design. Master's thesis, Universit della Svizzera Italiana (2000)

[10]Chambers, C., Harrison, B., y Vlissides, J. A debate on language and tool support for design patterns. In Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM Press (2000) 277–289
[11]Coplien. J. O. Software Design Patterns: Common Questions and Answers, pages 311–320. Cambridge University Press, New York (1998)
[12]Cybulski, J.L. y Linden, T. Pattern Languages of Program Design, volume 4, chapter Composing Multimedia Artefacts for Reuse. Addison-Wesley Longman (1999) 461–488
[13]Dearden, A., Finlay, J., Allgar, E. y McManus, B. Using pattern languages in participatory design. In Proceedings of the Participatory Design Conference (2002) 104 – 113
[14]Díaz, P., Aedo, I. y Panetsos, F. A methodological framework for the conceptual design of hypermedia systems. In Proc. of the fifth conference on Hypertexts and hypermedia: products, tools and methods (1999) 213–228
[15]van Duyne, D. K., Landay, J. A. y Hong, J. I. The Design of Sites:Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. Addison-Wesley (2002)
[16]Ebersole, S. Cognitive Issues in the Design and Development of Interactive Hypermedia: Implications for Authoring WWW Sites, Interpersonal Computingand Technology: An Electronic Journal for the 21st Century (1997) 19-36
[17]Escalona, M. J. y Koch, N. Requeriments engineering for web applications: A comparative study. Journal of Web Engineering, 2(3) (2003) 192–212
[18]Fernandez, E., y Pan, R. A Pattern Language for Security Models, Conference on Pattern Languages of Programming (2001)
[19]Fernández, E. B., Krishnakumar, R.N., Larrondo-Petrie, M.M. and Xu, Y. High-level Security Issues in Multimedia/Hypertext Systems. Communications and Multimedia Security II. P. Horster (ed.), Chapman & Hall (1996)13-24
[20]Fowler, M. Dealing with Roles. attern Languages Of Programming (PLOP'97) (1997)
[21]Gamma, E. , Helm, R., Johnson, R., y Vlissides, J. Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
[22]Garrido, A., Rossi, G. y Schwabe, D. Patterns systems for hypermedia. In Proceedings of The 3rd Pattern Language of programming Conference (1997)
[23]Garzotto, F., Paolini, P., Bolchini, D. y Valenti, S. Modeling-by-Patterns of web applications. In Advances in Conceptual Modeling: ER '99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling (1999) 293–306
[24]German, D. y Cowan, D. Towards a unified catalog of hypermedia design patterns. In Proceedings of 33rd Hawaii International Conference on SystemSciences, Maui, Hawaii (2000)
[25]Gomes, P., Pereira, F., Paiva, P., Seco, N., Carreiro, P., Ferreira, J.L. y Bento, C. Using CBR for automation of software design pattern. In Proceedings of the European Conference Case-Based Reasoning (2002) 534–548
[26]Graham, I. A Pattern Language for Web Usability. Addison Wesley Professional, 2003.
[27]Kappel, G., Retschitzegger, W. y Schwinger, W. Modeling customizable web applications - a requirement's perspective. In Kyoto International Conference on Digital Libraries (2000)
[28]Lowe, D. y Hall, W. Hypermedia and the Web: an engineering approach. Chichester: John Wiley and Sons (1999)
[29]Lucrédio, D., Alvaro, A., Santana de Almeida, E. y do Prado, A. F. Mvcase tool- working with design patterns. In SugarloafPLoP 2003 Conference (2003)
[30]Montero, S., Díaz, P. y Aedo, I. A Semantic Representation for Domain-Specific Patterns. The Third Metainformatics Symposium (2004)
[31]Montero, S., Díaz, P. y Aedo, I. Formalization of web design patterns using ontologies. In Proc. of First International Atlantic Web Intelligence Conference, AWIC, volume 2663 of LNCS (2003) 179–188

[32]Montero, S., Aedo, I. y Díaz, P. Generation of Personalized Web Courses Using RBAC. Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference (2002) 419-423
[33]Montero, S., Díaz, P. y Aedo, I. Requirements for hypermedia development methods: A survey of outstanding methods. In Proc. of Advanced InformationSystems Engineering, 14th International Conference, CAiSE, volume 2348 of LNCS (2002) 747–751
[34]Nanard, J. y Nanard, M. Toward an hypermedia design patterns space. In 2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia (1999)
[35]Nielsen, J. Designing Web Usability. New Riders Publishing (2000)
[36]Preece, J., Rogers, Y. y Sharp, H. Interaction Design: Beyond Human-Computer Interaction. New York, NY: John Wiley & Sons (2002)
[37]Resource Description Framework (RDF) http://www.w3.org/RDF/ (2004)
[38]Rossi G., Schwabe D., Danculovic J., Miaton L. Patterns for Personalized Web Applications. Proceedings of EuroPlop´01, Rüping A., Eckstein J., Schwanninger C. (Eds.) (2001) 423-436
[39]Rossi, G., Schwabe, D. and Guimaraes, M. Designing personalized web applications. In Proceedings of the World Wide WebConference (2001) 275–284
[40]Rossi, G., Schwabe, D. y Garrido, A. Design reuse in hypermedia application development. In Proceedings of the 8th. ACM Conference on Hypertext: Hypertext'97 (1997) 57 - 66
[41]Rossi, G., Garrido, A. y Carvalho, S. Design Patterns for Object-Oriented Hypermedia Applications. Pattern Languages of Programs II. Addison-Wesley (1996) 177 – 191
[42]Web Ontology Language (OWL). http://www.w3.org/2004/OWL/ (2004)
[43]van Welie, Martijn. Web design patterns. http://www.welie.com/patterns/index.html (2005)