UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

INTEGRACIÓN DE PATRONES EN EL PROCESO DE DISEÑO DE SISTEMAS HIPERMEDIA MEDIANTE EL USO DE ONTOLOGÍAS

TESIS DOCTORAL

Susana Montero Moreno
Leganés, Madrid, 2005

DEPARTAMENTO DE INFORMÁTICA

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INTEGRACIÓN DE PATRONES EN EL PROCESO DE DISEÑO DE SISTEMAS HIPERMEDIA MEDIANTE EL USO DE ONTOLOGÍAS

TESIS DOCTORAL

AUTOR: Susana Montero Moreno
DIRECTORES: Dra. María Paloma Díaz Pérez
Dr. Ignacio Aedo Cuevas

Leganés, Madrid, 2005

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Carlos III de Madrid, el día ..... de ..... de 2005.

Presidente: D. .....

Vocal: D. .....

Vocal: D. .....

Vocal: D. .....

Secretario: D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2005 en .....

Calificación: .....

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

A Rafa

Agradecimientos

Aunque la elaboración de una tesis doctoral es producto del tesón y del esfuerzo personal que uno realiza en solitario, aunque cueste entenderlo, el empujón final viene dado por el apoyo científico, moral, y afectivo de las personas que te rodean, y sin ellas, hoy no podría estar escribiendo estas líneas.

Gracias a mis directores de tesis, por sus consejos y sus pacientes correcciones (yo no podría haberlas aguantado), por su confianza en mi, a veces ciega, y por ir asfaltando los otros caminos que quedan por recorrer. No sé si lo podré cambiar, pero intentaré no dejarlo todo para el último momento.

También las cuestiones monetarias han tenido su importancia en la elaboración de este trabajo. *Gracias a los proyectos de investigación de Ariadne y ARCE* que me han permitido tener a Juanfran como fiel sufridor y artífice material de algunas de las herramientas software incluidas en este trabajo. *Gracias Juanfran*, por sacar tiempo de donde no tenías y estar siempre preparado para darme una respuesta.

A parte de tener ideas, uno tiene que probarlas. *Gracias a mis compañeros de grupo y alumnos de 5º curso de Informática* por prestaros a mis experimentos.

Pero como no sólo de ciencia vive la mujer, también he necesitado mis raciones de comprensión, cariño y pataletas.

Gracias a mis padres, por comprender mejor que yo la necesidad de esta obligación que ha impedido que disfrutemos de más momentos juntos y por darme ánimos y cariño.

Gracias a mis compañeros de batalla, por ser siempre fuente de ánimos y de diversión, haciendo lo imposible posible, sacar de un desierto un oasis donde recuperar fuerzas.

Gracias a los amigos, a los de siempre y a los de la carrera, por hacer ese seguimiento semanal tanto sobre mi estado de ánimo como de la tesis. Y ahora que no me escucháis, odio la frase “pero ..., ¿cuánto te queda?”.

Y finalmente a ti, *Rafa*, por tu apoyo, compresión y serenidad, por hacer del sinsentido una prueba de amor, y como cada minuto robado ha parecido contar más que ningún otro, esto te pertenece.

En fin, gracias por cada sonrisa y apoyo brindado.

Leganés, a 27 de mayo de 2005.
Susana M.

Resumen

El desarrollo de sistemas hipermedia y web a gran escala debe llevarse a cabo siguiendo un proceso sistemático y bien definido, es decir, aplicando métodos y técnicas específicos de la ingeniería de la hipermedia y de la web. Además, teniendo en cuenta que el proceso de construcción de sistemas web se caracteriza por el poco tiempo disponible para conseguir un producto operativo, los diseñadores pueden aliviar la toma de decisiones recurriendo a los patrones de diseño. Un patrón de diseño recoge 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. 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 de diseño hipermedia (pdh) debería, como punto de partida, buscar en las múltiples publicaciones existentes o en alguno de los repositorios web disponibles y ser capaz de identificar, por sus propios medios, aquel que mejor se ajusta a sus necesidades. Hasta la fecha, no existen los mecanismos necesarios para organizar los pdh de tal modo que ayuden al diseñador a encontrar el patrón o patrones apropiados y a integrarlos en los métodos de diseño. Por lo tanto, el éxito de esta actividad dependerá tan sólo del conocimiento que tenga el diseñador sobre los recursos existentes y su experiencia en el uso de patrones.

Para resolver estas carencias, el primer objetivo de esta tesis ha sido **organizar el conocimiento de los pdh**, proporcionando unos **criterios de clasificación** que tienen en cuenta tanto las necesidades del modelado de las aplicaciones hipermedia como los diferentes niveles de abstracción utilizados en la literatura de los pdh. A partir de esos criterios se ha elaborado un **catálogo de patrones** destinado a facilitar la búsqueda de patrones en la resolución de problemas concretos de diseño y un **lenguaje de patrones** orientado a ser una guía activa durante el desarrollo de una aplicación hipermedia, que aconseje al diseñador cómo acometer dicha empresa desde diferentes niveles de abstracción, a la vez que le propone otros problemas a considerar. Posteriormente, teniendo como base el lenguaje de patrones definido, se ha especificado un **proceso de integración con los métodos de diseño**, en el cual un conjunto de pasos guían una aproximación basada en patrones que permite transformar la especificación de requisitos en entidades de diseño en el modelado conceptual,

gracias a las soluciones capturadas por los patrones. Este proceso ha sido evaluado de forma empírica para comprobar su validez y utilidad.

Una vez que el espacio y el conocimiento de los patrones están organizados, se debían afrontar nuevos retos y realidades. En particular, la gran cantidad de patrones y sus diferentes fuentes, aún contando con mecanismos como los anteriormente mencionados, suponen a un diseñador no experimentado en el uso de patrones la necesidad de explorar la gran mayoría de ellos, además del esfuerzo de entender cuándo y cómo el patrón debe ser aplicado. Llegado este punto se pone de manifiesto la necesidad de contar con herramientas que permitan la organización, recuperación y exploración de manera automatizada, con funcionalidades de búsqueda dedicadas tanto a la recuperación de patrones con respecto a problemas de diseño genéricos como su aplicación en otros sistemas software, como pueda ser el caso de las herramientas CASE. Para contar con este tipo de herramientas y que además, puedan ser utilizadas de manera conjunta, es necesaria una especificación formal del patrón. 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 que permita la interoperabilidad entre los diferentes métodos de diseño, y de las propuestas realizadas en el dominio de la ingeniería del software tan 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. Por ejemplo incluir la intención del patrón permitiría indexar los patrones según el problema que resuelven, ayudando a los usuarios a encontrar el patrón correcto para un problema de diseño.

Por ello, se ha propuesto un **modelo de representación semántico** basado en ontologías que combina tanto el conocimiento del dominio hipermedia utilizado para la descripción de los patrones como el formato del patrón. Posteriormente, esta representación es utilizada como armazón subyacente para complementar la representación textual del patrón mediante **anotaciones semánticas**, la cual es soportada por una herramienta que ayuda tanto a usuarios como a autores de patrones a formalizar sus propios repositorios. Esta combinación de representación formal enlazada a su representación textual mediante anotaciones hace a los patrones mutuamente comprensibles a diseñadores y a programas, siendo muestra de ello el desarrollo de un repositorio web semántico y la automatización del proceso de integración de los patrones en una herramienta de soporte al diseño de aplicaciones hipermedia, utilizando ambos desarrollos el lenguaje de patrones formalizado mediante la herramienta de anotaciones propuesta. El desarrollo de estas herramientas ha probado la factibilidad técnica de la solución aquí propuesta.

Abstract

The development of large-scale hypermedia and web systems must be carried out following a well defined systematic process, that is, applying hypermedia and web engineering methods and techniques. In addition, considering that the development process of web systems is characterized by obtaining an operative product in a short period of time, designers can alleviate decision making resorting to design patterns. A design pattern records the knowledge and the experience of domain experts on how to make a software system more reusable and flexible as a result of many efforts on the design and codification of systems. Currently, if a designer or a developer of hypermedia applications wants to solve or to communicate a specific design problem by means of hypermedia design patterns (hdp), as starting point, she would have to search across multiple existing publications or some of web repositories available in order to identify the pattern or patterns that best fit her needs. To date, the necessary mechanisms to help the designer to find the appropriate pattern or patterns and to integrate them into hypermedia design methods do not exist. Therefore, the success of this activity will depend only on the designer knowledge about the existing resources and her experience in the use of patterns.

In order to solve these lacks, the first aim of this thesis has been to **organize the knowledge of hdp**, providing a **set of classification criteria** which takes into account as much hypermedia application modeling needs as the different levels of abstraction used in the pdh literature. From these criteria, a **pattern catalogue** has been elaborated to facilitate the search of patterns during the resolution of particular design problems as well as a **pattern language** as a step-by-step guide that advises designers how to accomplish the design process from different levels of abstraction, whereas it proposes alternative problems to resolve. Finally, from the pattern language, an **integration framework with design methods** has been specified, in which a set of steps guides a pattern based approach to transform requirements specifications to design entities into the conceptual modeling, thanks to the solutions captured by patterns. This process has been evaluated in an empirical way to verify its validity and utility.

Once the space and the knowledge of patterns are organized, new challenges and realities should be faced up to. In particular, the great amount of patterns and its different sources, even counting on the mechanisms before mentioned, lead a non-experienced des-

igner in the use of patterns to the necessity to explore most of them, in addition to the effort to understand when and how the pattern must be applied. At this stage, it is worth counting on software tools that allow for the organization, recovery and exploration of patterns with searches dedicated to finding appropiate patterns to generic design problems as well as their application in other software systems, such as CASE tools. However, firstly it is necessary to share a formal specification of patterns in order to preserve compatibility among software tools. To date, the hypermedia engineering field does not have its own formalization that allows the development of software tools for patterns, and the formalizations proposed in the software engineering field only take into account the pattern solution and leave aside the textual pattern description, so important to understand the foundation that underlies pattern's problem and solution. All elements of a pattern are required to develop effective tools for patterns organization, recovery and exploration. For example, to include the intention of pattern would allow for indexing patterns according to the problem that solve, helping users to find the appropiate pattern for a design problem.

For that reason, a **semantic representation model** based on ontologies that combines the domain knowledge about hypermedia used for the pattern description with the pattern format has been proposed. Afterwards, this representation provides the underlying framework to complement the textual pattern representation by means of **semantic annotations**, which is supported by a tool that helps both users and authors of patterns to formalize their own repositories. This combination of the formal representation linked to the textual pattern representation by means of annotations allows to patterns to be mutually comprehensible by designers and programs. An example of this is the development of a semantic web repository and the automatization of the integration process of hdp in a tool support for the design of hypermedia applications, using both developments the pattern language formalized by means of the annotation tool proposed. The development of these tools have proven the technical feasibility of the solution here proposed.

“Hay que decir lo que hay que decir
pronto
de pronto
visceral del tronco;
Con las menos palabras posibles
que sean posibles los imposibles
hay que hablar poco y decir mucho
hay que hacer mucho
y que nos parezca poco”
- Gloria Fuertes -

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

> "*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.*"
>
> [Alexander *et al.*, 1977]

1.1. Contexto

En 1977, Christopher Alexander, de profesión arquitecto, publica su libro *"A Pattern Language"* [Alexander *et al.*, 1977] en el que recopila un conjunto de patrones cuya combinación permitiría a cualquier persona diseñar y construir su propio entorno, introduciendo la metáfora de *patrón* como una solución definitiva a un problema que se repite.

Para ello, los patrones de diseño, al igual que los cuentos, van relatando al lector su propia historia, es decir, el contexto en el cual el patrón puede ser aplicado, el problema y los motivos que llevaron a su existencia, cómo los elementos que lo forman colaboran para proporcionar una solución reutilizable, y, finalmente, a modo de moraleja, describen las consecuencias de aplicar el patrón, permitiendo así evaluar al propio lector las alternativas de diseño a las que se puede enfrentar. Todo ello, es redactado en un lenguaje carente de tecnicismos y acompañado de ilustraciones que esquematizan la solución que propone el patrón,

convirtiendo así a los patrones en una herramienta de comunicación que permite llevar a cabo un proceso de diseño participativo entre arquitectos y habitantes [Alexander *et al.*, 1985].

Aunque dentro del área del urbanismo no tuvo un gran impacto, su idea de construir sistemas complejos mediante patrones fue acogida de manera entusiasta dentro de la comunidad de desarrollo de software tras la aparición del libro *"Design Patterns, Elements of Reusable Object-Oriented Software"* [Gamma *et al.*, 1994], donde sus autores recogían aquellas experiencias en la resolución de problemas de diseño de la orientación a objetos que habían sido probadas y usadas en muchos desarrollos, documentándolas con un formato estructurado y caracterizándolas por un nombre que transmitía su intención. Los patrones de diseño pretenden hacer explícito aquello que el experto hace muchas veces implícitamente cuando emplea procedimientos en un nivel muy abstracto para dar solución a problemas que aparecen recurrentemente, de tal manera que otros diseñadores o desarrolladores no tan expertos puedan beneficiarse de ese conocimiento y aplicarlo en otro dominio.

En los años siguientes a la publicación del *GoF*[1] se ha producido un auge en la investigación de patrones, extendiéndose a otras áreas del desarrollo software, incluyendo: patrones de análisis (conjunto de reglas que permiten modelar un sistema de forma satisfactoria) [Fowler, 1997a], patrones de procesos (serie de acciones para desarrollar software) [Ambler, 1998], patrones de arquitectura (formas de descomponer, conectar y relacionar sistemas) [Buschmann *et al.*, 2001] y patrones de programación (patrones de bajo nivel en un lenguaje de programación concreto) [Alur *et al.*, 2003]. Otras muchas áreas relacionadas con el software también han registrado sus mejores soluciones mediante el uso de patrones, como por ejemplo los sistemas distribuidos, concurrentes o paralelos [14][2], la interacción persona-ordenador [32], el comercio electrónico [Adams *et al.*, 2001], o la inteligencia artificial [Peters, 2003]. Además, el modelo descriptivo de los patrones ha demostrado también su efectividad en otras áreas no relacionadas con el software, como la organización de grupos humanos [Keidel, 1995].

Entre las bondades de los patrones de diseño se encuentran las de facilitar el diseño, mejorar la documentación, ayudar a reestructurar sistemas existentes e, incluso, proporcionar un vocabulario común que permita la comunicación de las diferentes alternativas de diseño de manera eficiente [Chambers *et al.*, 2000]. Sin embargo, con el crecimiento del número

[1] *Gang of Four*, nombre con el que se conoce familiarmente al libro *"Design Patterns, Elements of Reusable Object-Oriented Software"*

[2] Para una aclaración de los diferentes estilos de citas bibliográficas véase la sección 1.5.2 Convenciones tipográficas.

de patrones publicados, la acumulación de información evoluciona hacia una masa carente de estructura, lo que hace necesario organizar el conocimiento de los patrones mediante catálogos y lenguajes para conseguir un uso más efectivo. Un catálogo es un grupo de patrones clasificados por algún criterio y relacionados entre sí, los cuales pueden ser utilizados de forma conjunta o independiente, mientras que un lenguaje es una colección, que tiene una cohesión que revela las estructuras y las relaciones de sus componentes para cumplir un objetivo compartido [Noble, 1998]. Además, 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 [Lucrédio *et al.*, 2003], verificar su presencia [Albin-Amiot y Guéhéneuc, 2001], o encontrar los más adecuados a un determinado problema [Gomes *et al.*, 2002].

Mientras que en la comunidad de la orientación a objetos los patrones de diseño son escritos por y para ingenieros del software o personas con habilidades y conocimientos similares que los utilizan como mecanismo de reutilización del software, existen otras comunidades, también con una larga tradición, como es el caso de la comunidad de la interacción persona-ordenador[3] (HCI *Human Computer Interaction*) que vuelve a la idea original de Alexander que postulaba los patrones como herramienta de comunicación entre profesionales (arquitectos) y usuarios del sistema (habitantes). Concretamente, esta comunidad aboga por una *lingua franca*, no sólo formada por un vocabulario sino por un marco conceptual que permita la comunicación entre todos los miembros que tienen que cooperar en el diseño de los sistemas interactivos, es decir, diseñadores, usuarios y otros agentes involucrados (*stakeholders*) a través de un lenguaje de patrones [Erickson, 2000, Borchers, 2000].

Para finalizar con las bondades aquí expuestas sobre los patrones de diseño, también cabe destacar que han mostrado su utilidad pedagógica de diferentes maneras: capturando el conocimiento experto en la práctica de enseñar y aprender [25] o como recursos de aprendizaje para diferentes materias [Seffah, 2003, Porter y Calder, 2004].

[3] Desde 1997 hasta la fecha existe un *workshop* dedicado a este tipo de patrones en la conferencia anual CHI.

1.2. Motivación

En un breve periodo de tiempo, la demanda de sistemas hipermedia[4], y en particular las aplicaciones web[5], así como su tamaño y complejidad se ha incrementado de forma vertiginosa en diferentes áreas como la educación, el comercio electrónico, la documentación, etc.

Todas estas aplicaciones tienen como objetivo conseguir que el usuario pueda llevar a cabo algún tipo de tarea, como comprar o aprender algo, caracterizada porque el conocimiento de cómo realizar dicha tarea reside en el usuario, de tal manera que la aplicación debe responder de la forma que éste espera. Para ello, **la hipermedia proporciona sofisticadas estructuras de navegación, comportamientos interactivos y composiciones multimedia** que imprimen a estas aplicaciones un marcado carácter cognitivo y estético [Nanard y Nanard, 1995]. Últimamente, se está potenciando la accesibilidad a diferentes audiencias y dispositivos mediante la **personalización** de la información y de las funcionalidades, así como la **seguridad** para preservar la confidencialidad, la integridad y la disponibilidad de dicha información [Aedo *et al.*, 2003].

Es ampliamente reconocido que para hacer frente al desarrollo de este peculiar tipo de aplicaciones, especialmente aquellas de gran escala con una larga perspectiva de vida, debe llevarse a cabo un proceso sistemático y bien definido propio de una ingeniería [Baresi *et al.*, 2002]. La **ingeniería de la hipermedia** ha tenido en cuenta la experiencia adquirida en las disciplinas de la ingeniería del software [Lowe y Hall, 1999] y de la interacción [Preece *et al.*, 2002], dando como resultado el desarrollo de métodos propios para esta tecnología, como puedan ser RMM [Isakowitz *et al.*, 1998], OOHDM [Schwabe y Rossi, 1998], WebML [Ceri *et al.*, 2000], UWE [Hennicker y Koch, 2000], WSDM [De Troyer, 2001], OO-H [Gómez *et al.*, 2001] o ADM [Díaz *et al.*, 2001a], en los que las necesidades del usuario potencial o final guían el proceso de desarrollo y una serie de mecanismos permiten modelar las peculiares características de estas aplicaciones anteriormente mencionadas.

[4]El término hipermedia se suele diferenciar del término hipertexto en que mientras el primero se utiliza para referirse a aquellos sistemas que soportan diferentes contenidos multimedia, el segundo engloba a los sistemas sólo textuales. En este trabajo se utilizará el término hipermedia para hacer referencia a ambos sistemas.

[5]Las aplicaciones web se pueden considerar como una subclase de los sistemas hipermedia que utilizan una tecnología específica para gestionar la información pero que comparten la misma filosofía de acceso. En este documento se utilizará el término hipermedia como un concepto genérico que abarca todas las aplicaciones que contienen elementos multimedia en una estructura hipertextual, independientemente de la plataforma en la que estén implementadas.

Sin embargo, a pesar de contar con métodos de ingeniería que asisten en la tarea de construir un producto completo, correcto, útil y utilizable, los diseñadores deben afrontar todavía **dos cuestiones**. Por un lado, **el desarrollo e implantación de proyectos web se caracteriza por el poco tiempo disponible para conseguir un producto operativo** [Barry y Lang, 2001] y, además, **un diseñador, tanto si es experto como no, puede llegar a adoptar una solución inadecuada a un determinado problema de diseño** (v.g. desorientación, sobrecarga de información, pobre usabilidad, insatisfacción y carencia de confianza en el servicio ofrecido), que no cubra todas las necesidades del usuario, 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 (técnicos, de seguridad, de interacción, etc.) para los cuales a veces puede no poseer las habilidades necesarias [Bolchini, 2000]. Esto, unido al vertiginoso cambio de las tecnologías, sugiere la **necesidad de llevar a cabo un proceso de reutilización de diseños**, más que de unidades de implementación.

Los diseñadores pueden aliviar la toma de decisiones recurriendo a los **patrones de diseño**, a la vez que cuentan con soluciones contrastadas y validadas empíricamente que pueden ser integradas en sus diseños. Un patrón de diseño recoge 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, como se ha expuesto en la sección 1.1. Sin embargo, esta reutilización no es tan inmediata como pueda parecer, puesto que es preciso encontrar el patrón que mejor cubra el problema de diseño a resolver y adaptarlo al contexto donde se aplicará. Por esta razón es muy importante contar con **herramientas y mecanismos que asistan a los diseñadores** en estas tareas así como **en la integración de los patrones en el proceso de diseño basado en métodos** con el objeto de mejorar la productividad y la calidad de las aplicaciones.

Los patrones de diseño irrumpen en la comunidad de la hipermedia de la mano de Gustavo Rossi en 1996, siendo muchos los trabajos que le siguieron: [Garrido *et al.*, 1997], [Bernstein, 1998], [Lyardet *et al.*, 1999], [Rossi *et al.*, 2000b], [Bonura *et al.*, 2002], [van Duyne *et al.*, 2002] y [Amitay *et al.*, 2003]. Los patrones hipermedia publicados hasta la fecha, algunos de ellos en repositorios web [12], [33], se centran en aspectos referentes a cómo presentar la información u organizar el acceso a dicha información desde un punto de vista cognitivo y estético. Debido a ésto, los patrones hipermedia son descritos de manera conceptual, sin llegar a detalles de implementación y utilizando elementos propios de los métodos y modelos de diseño hipermedia. Una vez que el do-

minio 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) [Garrido *et al.*, 1997]. Posteriormente, se presentaron diferentes clasificaciones como las de [Nanard y Nanard, 1999, Garzotto *et al.*, 1999] pero sin los correspondientes patrones catalogados. Finalmente, en [German y Cowan, 2000] 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.

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 complejo. En primer lugar, el diseñador debe comprender el esquema de clasificación usado en el catálogo. En segundo lugar, debe buscar la parte del catálogo en la que 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 proporciona 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 [Coplien, 1998]. 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 [van Duyne *et al.*, 2002] u orientados a la usabilidad de la aplicación [Graham, 2003a]. Como última característica a destacar de 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 [Garzotto *et al.*, 1999], o bien, para enriquecer dichos métodos con primitivas de más alto nivel de abstracción [Rossi *et al.*, 1997], es decir la aplicación de un patrón, incluyendo los elementos de diseño afectados, podría ser representada en el método con una única notación como sucede en [Sunyé *et al.*, 2000] para el lenguaje de modelado de orientación a objetos UML [Rumbaugh *et al.*, 1998].

Por lo tanto, a partir de esta breve panorámica del estado actual de los patrones de diseño hipermedia se desprenden una serie de carencias que dificultan su utilización por diseñadores y desarrolladores durante el proceso de diseño de las aplicaciones hipermedia así como su integración con los métodos de diseño:

- No existen unos criterios que ayuden a organizar los patrones existentes diseminados entre publicaciones y repositorios web, de acuerdo con las características de las aplicaciones hipermedia y la diversidad de niveles de abstracción encontrados en los patrones.

- No existen herramientas que asistan al diseñador en el proceso de búsqueda y selección de un patrón o le guíen en la resolución de un problema de gran escala como el diseño de una aplicación hipermedia mediante la combinación de patrones.

- No existe un mecanismo detallado que permita integrar los patrones en el proceso de diseño de los métodos, a pesar del claro beneficio que supondría en la reutilización del diseño y que algunos de esos métodos promueven [Montero *et al.*, 2002].

- No existe una homogeneidad en el nivel de descripción de los patrones ni en el vocabulario utilizado que permita una representación no ambigua para su posible automatización. Por un lado, patrones como los recogidos en [Garrido *et al.*, 1997] y [Garzotto *et al.*, 1999] se caracterizan por promover la reutilización de macroestructuras de diseño dependientes del método o modelo del cual el autor del patrón es experto, pero podrían ser aplicados de manera automática en el proceso de diseño. En contraposición, los patrones recogidos en [van Duyne *et al.*, 2002] y [Graham, 2003a] están más orientados a la descripción de problemas de alto nivel de abstracción, más cercana a la visión que tiene el usuario final de una aplicación hipermedia, que aunque no pueden ser aplicados en los métodos de diseño, podrían ser automatizados para permitir su exploración.

Todas estas carencias se deben a que, en un primer momento todo el énfasis se puso en el descubrimiento de patrones que recogiesen las mejores prácticas de diseño del área, y pocos fueron los esfuerzos dirigidos a cómo se deberían de describir, clasificar, u organizar los patrones para que sirviesen de ayuda al diseñador en su proceso creativo, y menos aún en proporcionar un formalismo que permitiese el desarrollo de herramientas para el soporte de dichas actividades.

1.3. Objetivos y aportaciones

De lo expuesto en la sección anterior se desprende que el área de los patrones de diseño hipermedia carece del grado de madurez mostrado en otras áreas, como las de la ingeniería del software y la interacción persona-ordenador. En este sentido el esfuerzo de este trabajo no va dirigido al descubrimiento de nuevos patrones, sino a **proporcionar las bases cognitivas y formales que permitan la integración del conocimiento y la experiencia registrada en modo de patrones en el proceso de diseño de las aplicaciones hipermedia, con el objeto de que diseñadores y desarrolladores puedan ser asistidos tanto de forma manual como automatizada**.

Para alcanzar este objetivo general se plantean los siguientes objetivos específicos:

O1. Definir un **nivel cognitivo** que organice el conocimiento subyacente a los patrones de diseño hipermedia mediante un conjunto de herramientas que asistan al diseñador tanto en la búsqueda de un patrón según un problema concreto de diseño, como en la selección combinada de patrones para la resolución de un problema de gran escala, donde dichas herramientas puedan ser utilizadas tanto de manera independiente como en conjunción con los métodos de diseño según las necesidades del diseñador.

O2. Definir un **nivel conceptual** que permita formalizar la estructura y el conocimiento de los patrones de diseño hipermedia, proporcionando una representación común independiente de métodos y modelos, con el objeto de que los patrones puedan ser tratados computacionalmente, tanto para su exploración como aplicación.

O3. Definir un **nivel de implementación** que demuestre que el nivel conceptual definido permite el desarrollo de repositorios para la gestión, organización y recuperación de los patrones de manera compartida por herramientas software que automaticen las definidas en el nivel cognitivo.

Todos estos objetivos se han materializado en:

- Un **espacio de categorización** que permite clasificar los patrones de diseño hipermedia de acuerdo con la naturaleza del problema en el que serán aplicados, el modelado conceptual de las aplicaciones hipermedia, y el nivel de abstracción en el que es descrito dicho problema. A partir de dicho espacio se ha seleccionado un conjunto de

patrones para formar parte de un **catálogo**, que asiste al diseñador en la búsqueda del patrón que resuelve un problema específico de diseño en un nivel de descripción determinado y un **lenguaje** que aconseja al diseñador cómo acometer el diseño de una aplicación hipermedia mediante la combinación de patrones en diferentes niveles de abstracción. Posteriormente, estas herramientas han servido de base para la definición de un **proceso de integración con los métodos de diseño**, donde los requisitos guían una aproximación basada en patrones para afrontar el modelado conceptual [Montero *et al.*, 2005].

- Un **modelo de representación** basado en ontologías que define formalmente tanto la estructura como el conocimiento del dominio hipermedia utilizado para la descripción de los patrones [Montero *et al.*, 2003a]. La representación formal del patrón es fruto de la asociación entre aquellos elementos encontrados en la descripción textual del patrón que recogen su esencia y su correspondiente significado semántico. Este vinculo entre descripción textual y formal se mantiene para permitir así su interpretación, tanto por diseñadores como por programas software.

- Una **herramienta de anotaciones**, llamada AnnotPat [Montero *et al.*, 2004b], que ayuda al diseñador a gestionar sus propios repositorios de patrones descritos según el modelo de representación formal. Dicha herramienta se ha utilizado para formalizar el lenguaje de patrones de diseño hipermedia a partir del cual se ha generado un **repositorio web**, que permite la exploración de los patrones mediante búsquedas avanzadas, y se ha integrado en una **herramienta CASE** [Montero *et al.*, 2004a] para la generación de aproximaciones al modelado conceptual siguiendo el proceso de integración definido. Ambas implementaciones demuestran que el modelo de representación propuesto permite la automatización de los patrones tanto para su exploración como para su aplicación.

Todo lo aquí expuesto será debidamente desarrollado y justificado en los diferentes capítulos que conforman esta tesis, como se mostrará en las siguientes secciones.

Esta tesis doctoral se ha realizado en el marco de dos proyectos de investigación en los que la doctoranda ha participado, y sigue haciéndolo, activamente. El primero de ellos es "Sistema de Ayuda para la Construcción de Aplicaciones Hipermedia/Web Basado en el Uso de Patrones de Diseño" que fue financiado por la Dirección General de Investigación de la Comunidad de Madrid y que tiene como referencia 07T/0024/2003. El segundo de ellos es

“ARCE++: Métodos avanzados de acceso a un sistema de información para la cooperación internacional en situaciones de desastre” que está financiado por el Ministerio de Educación y Ciencia con la referencia TSI2004-03394.

1.4. Metodología de la investigación

El procedimiento que se empleará para llevar a cabo el propósito de esta tesis se puede plasmar en los siguientes pasos [Sierra, 1986]:

1. Determinación del problema:
   - *a*) Definición del contexto del problema y los motivos por los cuáles surge este trabajo (el presente capítulo).
   - *b*) Elaboración de un estado de la cuestión sobre la teoría y conocimientos sobre el tema y las posibles investigaciones realizadas anteriormente con relación al mismo (véase el capítulo 2).
   - *c*) Definición del problema de una manera concreta y explícita (véase el capítulo 3).
2. Formulación de la hipótesis:
   - *a*) Elaboración de unos objetivos a alcanzar con la realización de este trabajo (véase el capítulo 3).
   - *b*) Diseño de una solución al problema que pueda ser verificable, tanto empírica como cualitativamente (véanse los capítulos 4 y 5).
3. Comprobación de la hipótesis:
   - *a*) Aplicación de la solución a casos de estudios concretos (véase el capítulo 6).
   - *b*) Determinación de la forma concreta de realizar la verificación de dichos casos de estudio (véase el capítulo 6).
4. Análisis de resultados:
   - *a*) Determinación de la significación y el alcance del análisis realizado en los casos de estudio y compararlos con los objetivos de partida (véase el capítulo 7).

b) Generalización de los resultados y su validez en otras áreas relacionadas (véase el capítulo 7).

1.5. Estructura del documento

Los siguientes apartados recogen brevemente tanto el contenido de cada uno de los capítulos de los que consta este documento como una serie de aclaraciones sobre las convenciones tipográficas utilizadas en su escritura.

1.5.1. Organización

El presente documento consta de los capítulos que a continuación se detallan:

Capítulo 1. Introducción. Es el presente capítulo, en el cual se introduce el contexto y la motivación que llevan al desarrollo de esta tesis, con la exposición final de una serie de objetivos a desarrollar. Además, se detalla la metodología de investigación a seguir y ciertas aclaraciones sobre las convenciones tipográficas utilizadas en este documento.

Capítulo 2. Estado de la cuestión. Es el capítulo que recoge una breve descripción de los temas clave para la realización de este trabajo como son: los patrones de diseño, tanto desde la perspectiva de la ingeniería del software, como del área de la hipermedia y su relación con el proceso de diseño de las aplicaciones hipermedia.

Capítulo 3. Planteamiento del problema. Con este capítulo se hace una valoración del estado de la cuestión y se presentan los problemas y carencias encontrados durante dicho estudio que sirve de justificación para el trabajo que a continuación se va a presentar.

Capítulo 4. Integración de los patrones en el proceso de diseño hipermedia. Este capítulo muestra cómo se ha organizado el espacio de los patrones hipermedia y los diferentes mecanismos que se proponen tanto para utilizar los patrones de forma individual como en conjunción con los métodos de diseño.

Capítulo 5. Conceptualización de los patrones hipermedia. La segunda parte de este trabajo, que es recogida en este capítulo, presenta un modelo de representación para

la formalización semántica de los patrones hipermedia que tiene como objetivo sentar las bases para el desarrollo de herramientas de soporte para su organización, consulta, exploración, u otros fines.

Capítulo 6. Evaluación. Este capítulo recoge tanto los casos de aplicación, que servirán de estudio para validar las propuestas presentadas en los capítulos anteriores, como las diferentes evaluaciones llevadas a cabo para recoger los datos empíricos que justifican su utilidad.

Capítulo 7. Conclusiones. Realizada la evaluación del presente trabajo, en este capítulo se analiza el grado de cumplimiento de los objetivos y se enumeran las conclusiones obtenidas tras la elaboración de esta tesis, a la vez que se resumen las aportaciones realizadas, así como las posibles aplicaciones en otros campos y líneas de investigación abiertas.

La bibliografía que se referencia en este documento y los anexos que aportan información adicional sobre el trabajo realizado cierran el presente documento.

1.5.2. Convenciones

A continuación se exponen una serie de aclaraciones del estilo que ha sido utilizado en la redacción de esta tesis, facilitando así su comprensión:

- Las citas textuales irán en su idioma original, encerradas entre dobles comillas(“”), con letra *cursiva* y debidamente sangradas, incluyendo la referencia bibliográfica de donde fueron tomadas.

- Los términos que se mantengan en su idioma original, bien por ser aceptados como parte de la jerga informática o para preservar su semántica, aparecerán en *cursiva*.

- Cuando se haga referencia a un patrón de diseño, su nombre aparecerá con la fuente de `máquina de escribir` y en su idioma original, para no desvirtuar el sentido que quería transmitir su autor, además de preservar la manera en que los diseñadores se comunican entre sí soluciones de diseño basadas en patrones.

- Se han empleado dos estilos diferentes de citas bibliográficas, el sistema autor-año para aquellas referencias que se encuentran en fuentes publicadas en medios formales, con ISBN o ISSN, y el sistema numérico para hacer referencia a recursos web.
- Las notas de pie de página se han reservado para hacer aclaraciones sobre el sentido en que ciertos términos son utilizados en este trabajo.

Capítulo 2

Estado de la cuestión

La complejidad actual de las aplicaciones hipermedia, el breve periodo de tiempo impuesto por el mercado para el desarrollo de estas aplicaciones y el vertiginoso cambio de las tecnologías, sugieren la necesidad de llevar a cabo un proceso de reutilización de diseños, más que de unidades de implementación, que permita mejorar la productividad, en la medida en que el equipo de desarrollo cuente con soluciones directamente integrables en sus diseños, como la calidad de las aplicaciones, siempre que dichos diseños reutilizables recojan soluciones contrastadas y validadas en esa comunidad de desarrollo. Este último aspecto es una característica básica de los patrones de diseño. Ambos aspectos son cualidades que se pueden encontrar en los patrones de diseño, los cuales fueron acogidos por la comunidad software como un medio para capturar y comunicar mediante un formato sencillo y entendible el conocimiento experto en el desarrollo de aplicaciones con el objeto de ser reutilizado en múltiples contextos. Sus probados beneficios han hecho que otros dominios software también los adopten. Así por ejemplo, los patrones de diseño hipermedia y web recogen aquellos problemas de diseño recurrentes en la navegación, la estructura, la interacción o la interfaz de tal forma que se convierten en candidatos ideales para ser integrados en el proceso de desarrollo de este tipo de sistemas.

Este capítulo ofrece un recorrido por el estado de la cuestión de los patrones de diseño, incluyendo los trabajos más relevantes que conforman el marco teórico de esta tesis. En primer lugar, se presenta una panorámica del concepto de patrón de diseño desde sus orígenes en el área de la arquitectura urbanística, hacia su paso a la comunidad software de orientación a objetos y su posterior difusión a una gran diversidad de dominios tanto software

como no (véase la sección 2.1). Uno de estos dominios software es el área de la ingeniería hipermedia, en la cual se utiliza un enfoque sistemático basado en modelos y métodos para el desarrollo de sistemas hipermedia y web (véase la sección 2.2). Dentro de este marco, los patrones de diseño hipermedia (pdh) han sido propuesto como una técnica complementaria a los métodos de diseño para aliviar al diseñador tanto en el acometido de nuevos desarrollos como en la toma de decisiones (véase la sección 2.3), sin embargo el estudio realizado sobre el estado actual de los pdh arroja una serie de carencias y problemas que pueden dar la clave de su poca aceptación dentro del área hipermedia, y que serán enumerados en el capítulo 3.

2.1. Patrones de diseño

Los patrones de diseño 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 en un formato simple y comprensible tanto para expertos como para noveles, sino que posibilitan la reutilización de la misma experiencia una y otra vez en diferentes aplicaciones.

Antes de profundizar en el concepto de patrones de diseño hipermedia, marco de esta tesis, y entender las carencias y los límites que llevan a la motivación de este trabajo, es necesario abordar el origen y la evolución de los patrones de diseño. Para ello, en esta sección se hará primeramente un repaso a las principales claves que subyacen en el origen de la metáfora de patrón dentro del dominio de la arquitectura urbanística (véase la sección 2.1.1), así como su adopción por la comunidad de la orientación a objetos para convertirse en el estandarte de la reutilización software y plataforma de expansión a todos los niveles de la ingeniería del software (véase la sección 2.1.2). Los patrones software pretenden ser el manual del ingeniero a la hora de resolver problemas de diseño, por ello, a medida que su número crece es necesaria su organización mediante catálogos (véase la sección 2.1.3) o lenguajes (véase la sección 2.1.4), tanto de manera manual como automatizada para ayudar al diseñador a la selección y aplicación de un patrón dado un problema de diseño específico (véase la sección 2.1.7). Finalmente, para concluir esta sección, se mostrará una serie de dominios software donde los patrones han sido también adoptados, pero los cuales son dependientes del dominio, como sucede con los pdh, en contraposición de los patrones de la orientación a objetos que son independientes del dominio (véase la sección 2.1.5) y es-

pecialmente se profundizará en los patrones de diseño de interacción persona-ordenador (véase la sección 2.1.6) por su especial relevancia en la actualidad y ser un dominio yuxtapuesto al dominio hipermedia que puede servir de marco de referencia para medir el estado actual de los pdh.

2.1.1. El origen de los patrones

En los años 70, el arquitecto Christopher Alexander se preguntó si había una base objetiva para medir la calidad de un diseño arquitectónico, es decir, si se podía identificar lo que hace a un diseño ser bueno o malo. Tras observar edificios, ciudades, calles, etc., descubrió que las construcciones de calidad tenían cosas en común. En 1977, en el libro *A Pattern Language* presenta junto a otros colegas la teoría y el uso de la metáfora de *patrón*:

> "*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*"
>
> [Alexander *et al.*, 1977]

La metáfora de patrón se puede resumir como un mecanismo para capturar la esencia de **una solución a un problema recurrente** en el diseño arquitectónico. Los patrones no son creados ni inventados, son identificados por un principio invariante que se manifiesta a través de diferentes sitios y culturas. Posteriormente, en su libro *The Timeless Way of Building* [Alexander, 1979], se explica como una colección jerárquica de patrones de diseño arquitectónicos se pueden identificar con objeto de hacer las construcciones más "usables" y agradables a sus habitantes, mejorando su confort y calidad de vida.

El objeto subyacente a los patrones de Alexander era el de capturar la experiencia y hacerla accesible a los profanos en la materia, para que pudiesen crear sus propias estructuras, desde la selección del emplazamiento de un edificio hasta la distribución de una habitación, dotándoles para ello del vocabulario necesario para expresar sus ideas y diseños y así poder comunicarse con los profesionales. Para ello organizó sus patrones en lo que llamó un *lenguaje de patrones* y los utilizó como una herramienta práctica para ayudar en procesos de diseño participativo entre arquitectos y usuarios [Alexander *et al.*, 1985]. Dicho lenguaje fue organizado en tres diferentes planos (basándose en las relaciones existentes entre los

patrones que formaban el lenguaje): planificación de ciudades, arquitectura y construcción y decoración de interiores.

Este concepto de patrón es tan genérico que dos décadas después fue acogido con un rotundo éxito dentro de la ingeniería del software tras la publicación del libro *Design Patterns: Elements of Reusable Object-Oriented Software* [Gamma *et al.*, 1994]. Este libro sirvió para asentar las bases de los patrones de diseño en la comunidad software y convertir a los patrones de diseño en un *hot topic* para los desarrolladores, profesionales e investigadores, dando lugar a diferentes conferencias anuales (v.g. la serie PLOP [26]) y grupos de estudio en diferentes áreas (*Hillside Group* [19], *HCI Patterns Task Group* [4] o *Security Patterns Community* [6]). Se puede decir que su contribución a la *reutilización* del software ya ha hecho historia.

2.1.2. Patrones de diseño en la ingeniería del software

Aunque dentro de la comunidad de desarrollo software ya se había empezado a utilizar el concepto de patrón, es con la publicación del catálogo de patrones recogido en [Gamma *et al.*, 1994] cuando se produce la verdadera revolución. Sus autores se refieren al libro como una recopilación de veintitrés patrones que describen soluciones simples y elegantes a problemas específicos en el diseño software orientado a objetos. El objetivo fundamental de este catálogo es el de ser compartidos entre los expertos de este paradigma.

Desde entonces, otros tipos de patrones software han sido publicados: patrones de análisis (conjunto de reglas que permiten modelar un sistema de forma satisfactoria) [Fowler, 1997a], patrones de procesos (serie de acciones para desarrollar software) [Ambler, 1998], patrones de arquitectura (formas de descomponer, conectar y relacionar sistemas) [Buschmann *et al.*, 2001] y patrones de programación (patrones de bajo nivel acerca de un lenguaje de programación concreto) [Alur *et al.*, 2003].

A medida que ha aumentado el uso de los patrones se ha tenido certeza de que éstos pueden mejorar la productividad y calidad de las soluciones software gracias a las siguientes cualidades [Chambers *et al.*, 2000]:

- **Promueven la reutilización del diseño**. Los patrones de diseño capturan experiencias en la resolución de problemas de diseño cuya utilidad o necesidad se ha eviden-

ciado repetidamente en diferentes proyectos. Por lo tanto, la aplicación de un patrón lleva implícita la reutilización de esos diseños.

- **Forman un vocabulario común de diseño**. Los nombres de los patrones ayudan a los diseñadores a comunicarse de una manera más eficiente, extendiendo su vocabulario y siendo capaces de transmitir el problema y la solución como un todo comprensible.

- **Mejoran la documentación**. La utilización de un patrón en el desarrollo de un sistema permite conocer qué solución se ha aplicado y por qué, puesto que el patrón no es una idea del programador sino una solución probada y reconocida en una comunidad de desarrollo.

Teniendo en cuenta que una de las cualidades principales de los patrones es la de comunicar soluciones reutilizables a otros diseñadores, un factor crítico es cómo realizar dicha descripción de tal manera que el destinatario sea capaz de comprender la motivación, la esencia y la utilidad del patrón. Para ello, los patrones de diseño, al igual que los cuentos, van relatando al lector su propia historia, es decir, el contexto en el cual el patrón puede ser aplicado, el problema y los motivos que llevaron a su creación, cómo los elementos que lo forman colaboran para proporcionar una solución reutilizable, y finalmente, a modo de moraleja, describen las consecuencias de aplicar el patrón, permitiendo así evaluar al propio lector las alternativas de diseño a las que se puede enfrentar.

En la literatura, siguiendo en mayor o menor medida la secuencia anteriormente descrita, podemos encontrar diferentes formatos para describir un patrón según el nivel de detalle con el que se quiera hacer la descripción [Rising, 2003]. El formato descrito en [Gamma *et al.*, 1994] y conocido como el formato "*GOF*" es el más completo y detallado, haciendo casi directa la implementación software del patrón. Otro de los formatos más usado es el propuesto por Alexander en [Alexander *et al.*, 1977], el cual describe la solución del patrón de forma más abstracta, mostrando un dibujo que representa dicha solución y permitiendo que sea implementado de diferentes maneras. En la figura 2.1 queda representada la estructura conceptual de un patrón de diseño mediante las relaciones existentes entre las diferentes partes que lo componen.

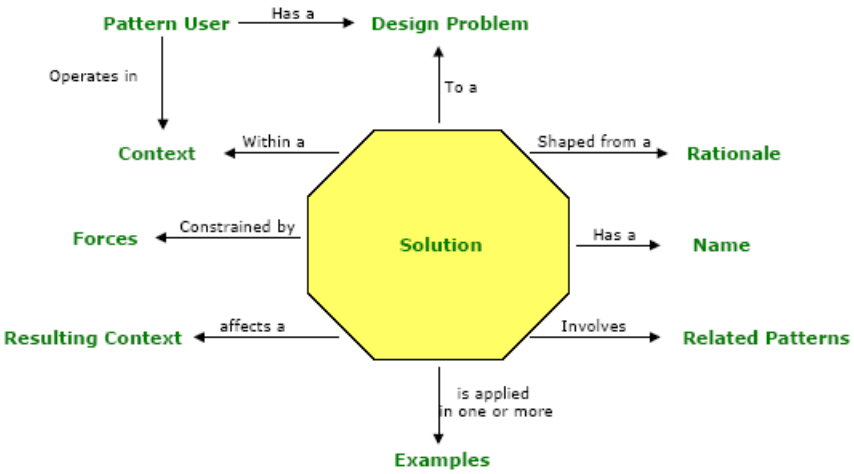


Figura 2.1: Estructura conceptual de un patrón [Bolchini, 2000]

2.1.3. Catálogos de patrones

Cada patrón representa en sí mismo un plan general que puede ser usado en un proceso de diseño para resolver un problema específico. Sin embargo para hacer frente a problemas complejos, los patrones deben estar recopilados en catálogos. El catálogo más conocido es el publicado en [Gamma *et al.*, 1994], pero existen otros, como el de arquitectura software [Buschmann *et al.*, 2001] o el de análisis [Fowler, 1997a].

A medida que el número de patrones de diseño crece, es necesario organizarlos 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. Las técnicas de clasificación de patrones permiten organizar los patrones en grupos que comparten el mismo conjunto de propiedades, y dependiendo de los criterios elegidos se pueden definir esquemas de clasificación con diferentes dimensiones. Los esquemas de clasificación ponen de manifiesto las principales cualidades de los patrones y ayudan a reducir el tamaño del espacio de búsqueda.

En la literatura se pueden encontrar los siguientes criterios utilizados para clasificar diferentes tipos de patrones [Kardell, 1997]:

- **Por su dominio**. Dentro del campo de la ingeniería del software, los patrones no están restringidos a resolver problemas de diseño o de arquitectura software. Considerando el problema del diseño, gran cantidad de patrones han sido propuestos en diferentes áreas, como puedan ser los de sistemas distribuidos, concurrentes o paralelos [14] o colaborativas [Guerrero y Fuller, 2001].

- **Por su paradigma**. Aunque los patrones son una práctica popular en el desarrollo de los sistemas orientados a objetos, no son exclusivos de este paradigma. En [Antoy y Hanus, 2002] se presenta un catálogo de patrones para lenguajes de lógica funcional, expresados en términos de programación declarativa como funciones y predicados. De igual manera sucede con los patrones hipermedia que están basados en el paradigma del hipertexto, como posteriormente se explicará.

- **Por su granularidad**. Los patrones también pueden ser clasificados dependiendo del nivel al cual se dirijan en el sistema. La clasificación más famosa es la descrita en [Buschmann *et al.*, 1996], donde en el nivel más bajo se encuentran los patrones específicos del lenguaje de programación, el nivel medio se ubican los de diseño y en el nivel superior los patrones referentes a la arquitectura del sistema.

- **Por su nivel de abstracción**. De una manera similar a la anterior, los patrones software se pueden clasificar en patrones conceptuales, los cuales son descritos mediante términos y conceptos de un dominio de aplicación específico; en patrones de diseño que describen construcciones software, complementando el espacio conceptual abierto por los patrones conceptuales; y en el nivel más bajo de abstracción estarían los patrones de programación que son descritos mediante construcciones propias de los lenguajes de programación [Riehle y Zullighoven's, 1996].

- **Por su propósito**. El propósito representa qué tipo de problemas resuelve el patrón. Este criterio fue usado inicialmente en [Gamma *et al.*, 1994], y más tarde en [Buschmann *et al.*, 1996].

- **Por su alcance**. Dentro del mismo paradigma y usando el mismo entorno de desarrollo, los patrones pueden ser clasificados según el espacio del entorno en el que se aplica la solución del patrón. En el paradigma de la orientación a objetos el patrón puede ser aplicado a una clase o a un objeto [Gamma *et al.*, 1994].

- **Por las relaciones entre patrones**. Los patrones propuestos en [Gamma *et al.*, 1994] especifican las relaciones existentes entre los patrones del propio catálogo. En

> [Zimmer, 1995] se examinaron la naturaleza de dichas relaciones clasificándolas en el patrón X usa el patrón Y en su solución, el patrón X es similar al patrón Y, y el patrón X puede ser combinado con el patrón Y, que posteriormente fueron utilizadas como esquema de clasificación para dicho catálogo.

Estos criterios, tanto de manera individual como en combinación permitirían definir esquemas de clasificación, por ejemplo en [Gamma *et al.*, 1994] se combinan los criterios **propósito** y **alcance** para classificar los patrones recogidos en su catálogo, y dado que los patrones poseen diferentes propiedades, el mismo patrón podrá ser incluido en diferentes esquemas.

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 al nivel del problema que se está intentando resolver. Además, estas relaciones no forman parte de la estructura del catálogo [Noble, 1998].

2.1.4. Los lenguajes de patrones

Alexander resolvió el problema de la selección de un patrón organizando sus patrones en lo que denominó un *lenguaje de patrones*. Este lenguaje tiene un vocabulario, que son los patrones en sí, y una gramática, que son las relaciones existentes entre los patrones. Así se forma una red donde cada patrón depende de los patrones de menor escala que contiene y de los patrones de mayor escala donde está contenido, permitiendo asistir al diseñador en la selección de los patrones que solucionan un determinado problema de diseño.

En realidad, el lenguaje de patrones presentado por Alexander es un meta-lenguaje, ya que tanto los patrones individuales como el propio lenguaje son maleables y pueden ser usados para generar lenguajes de patrones específicos para un escenario arquitectónico concreto.

En la terminología de Alexander, atravesar el lenguaje de patrones genera un diseño concreto, de modo que se proponen los siguientes pasos para llevar a cabo la selección del conjunto de patrones que resuelven un determinado problema [Alexander *et al.*, 1977]:

1. Elegir el patrón que mejor describe el alcance global del proyecto.
2. Ir al final del patrón, donde se referencian los patrones de menor escala que soportan ese patrón, y hacer una lista con los que parecen que se pueden aplicar al proyecto.
3. Para cada patrón seleccionado en el paso 2, repetir el paso anterior y también examinar los patrones de mayor escala que aparecen al comienzo de cada patrón, añadiendo todos los que sean relevantes a la lista.
4. Repetir los pasos 2 y 3 hasta conseguir una lista de patrones.
5. Ajustar la lista de patrones añadiendo material propio, modificando los patrones existentes para que sean más relevantes a la situación actual o creando nuevos patrones.

A la hora de afrontar la elaboración de un lenguaje de patrones, en una primera aproximación se recopilarían los patrones existentes en un catálogo de patrones siguiendo, o no, alguno de los criterios de clasificación expuestos en la sección anterior, proporcionando así una colección de patrones que resuelven una serie de problemas específicos. Posteriormente, se pasaría a estudiar las relaciones existente entre los patrones y se organizarían como un todo, proporcionando 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 [Coplien, 1998]. La característica que permite discernir si se está ante un lenguaje o un mero catálogo de patrones es la calidad y la naturaleza de las conexiones existentes entre sus patrones [Salingaros, 2000], por lo tanto es uno de los puntos críticos en la composición de un lenguaje. En la literatura pueden distinguirse dos tipos de relaciones, las basadas en las relaciones estructurales entre los patrones (p.e. composición, especialización, combinación, etc.) que disponen el lenguaje de una manera jerárquica [Zimmer, 1995, Tahvildari y Kontogiannis, 2002]; y las basadas en los requerimientos no funcionales presentes en la descripción de los patrones (p.e. tiempo, espacio, computación, dinero, esfuerzo, etc.) que permiten evaluar las ventajas y desventajas de utilizar un determinado patrón [Gross y Yu, 2000].

En general, la estructura de un lenguaje de patrones tendría la forma de un grafo dirigido, generalmente con pocos ciclos, donde los nodos representan patrones y los enlaces las

relaciones entre patrones. Un patrón inicial en la raíz del grafo esboza la solución general que proporciona el lenguaje al problema de gran escala que está abordando. Este patrón está relacionado con otros patrones de menor escala, que a su vez proporcionan soluciones, exponiendo más subproblemas que usan patrones de menor escala para resolverlos.

En la ingeniería del software todavía no se ha llegado a la riqueza lingüística expresada por el lenguaje de patrones de Alexander, hablando en términos de patrones, aunque ha habido varias aproximaciones en diferentes dominios. En [Noble, 1998] se recopilan noventa patrones de diseño orientados a objetos organizados en tres fragmentos según la clasificación realizada en [Buschmann *et al.*, 1996] (arquitectónicos, diseño, programación), y proporcionando tres tipos de relaciones (de uso, de conflicto y de refinamiento) para relacionar los patrones dentro de cada uno de esos fragmentos.

Finalmente, aunque los lenguajes de patrones son un valor añadido a la hora de enfrentarse a la utilización real de patrones durante el desarrollo de una aplicación software, la inherente ambigüedad ligada a su descripción textual y la dificultad de su aprendizaje han hecho que surjan diferentes aproximaciones para describir formalmente los patrones de diseño, y facilitar el desarrollo de herramientas de soporte para su aplicación.

2.1.5. Patrones específicos del dominio

El campo de aplicación de los patrones es extremadamente amplio, como ya se ha mencionando anteriormente. Todo comienza en el área de la arquitectura y, posteriormente, la misma metáfora ha sido aplicada en otras disciplinas como la ingeniería del software [Gamma *et al.*, 1994], la economía sostenible [15] o la enseñanza [25]. Dentro de la ingeniería del software, además de la comunidad de la orientación a objetos, existen diferentes áreas que afrontan los problemas de diseño capturando su conocimiento con patrones, y por lo tanto, estos patrones pueden ser reutilizados para diseñar aplicaciones que comparten dicho conocimiento. Este tipo de patrones reciben el nombre de **Patrones Específicos del Dominio**, pero rara vez sus autores usan este término para referirse a ellos, sino simplemente se les denomina patrones de diseño. Antes de reseñar las principales diferencias entre ambos tipos de patrones, se presentan una serie de dominios de conocimiento en los cuales los patrones cuentan con una literatura importante.

- **Interacción hombre-máquina.** Estos patrones están orientados a solventar los problemas que los usuarios finales encuentran cuando interactúan con sistemas, siendo la

usabilidad la cualidad esencial que se pretende primar en el diseño. Las colecciones de patrones de interacción se encuentran disponibles en libros [Borchers, 2001] o en web [32] con un total de más de 250 patrones publicados. A modo de ejemplo, el patrón `Progress Indicator` [32] debería ser utilizado cuando el usuario necesita saber si una operación está siendo todavía procesada por el sistema y cuánto tiempo falta para que finalice, pues el patrón proporciona un indicador válido de cómo está progresando dicha operación. Ejemplos de aplicación de dicho patrón serían el temporizador del microondas, la barra de progreso en entornos de escritorio gráficos o el mensaje de porcentaje completado en un navegador durante una descarga).

- **Seguridad.** Son patrones que dan soluciones a problema recurrentes de seguridad cuyo objetivo es permitir que los desarrolladores puedan tomar decisiones bien informadas llegando a un compromiso entre la seguridad y otras metas. Estos patrones incluyen diferentes aspectos de seguridad como modelos, aplicaciones o criptografía [Schumacher, 2003]. Al igual que en el caso anterior existen catálogos disponibles en libros [Blakley y Heath, 2004] o en web [6]. Un ejemplo de patrón es el `Check Point`[Konrad *et al.*, 2003] que propone evitar accesos no autorizados comprobando todas las peticiones de entrada al sistema y determinar si se permite o no el acceso. Ejemplos de aplicación de este patrón son la aplicación de entrada a Windows NT o un cortafuegos.

- **Inteligencia Artificial.** Dentro de este dominio pueden encontrarse patrones que capturan las prácticas de diseño comunes en el desarrollo de sistemas inteligentes [Peters, 2003], programación genética [Lenaerts y Manderick, 1998] o sistemas basados en agentes [Aridor y Lange, 2003], los cuales cuentan con una mayor literatura. Un ejemplo de estos últimos es el patrón `Agent Society` que propone modelar una aplicación como una sociedad de agentes si el dominio de dicha aplicación cumple con alguno de los siguientes requisitos: los datos, el control, el conocimiento o los recursos son centralizados. Ejemplos de aplicación de este patrón se pueden encontrar en aplicaciones de control, de trabajo en grupo o de comercio electrónico.

Los patrones de diseño de la orientación a objetos son denominados independientes del dominio porque son descritos de manera muy abstracta en términos de objetos y componentes, los cuales pueden ser aplicados a diferentes dominios adaptándolos al contexto. Por ejemplo el patrón `Observer` [Gamma *et al.*, 1994] ha sido adaptado en el patrón `Node as`

`a Navigational View` para el dominio hipermedia, del cual se hablará detalladamente en la sección 2.3.

Los patrones específicos del dominio son patrones, en el sentido original definido por Alexander, aplicados a diferentes aspectos del dominio software. Debido a la popularidad de los patrones de diseño de la orientación a objetos, la suposición natural es que cualquier patrón de diseño debería ser descrito en términos de clases y objetos, incluyendo un diagrama de la solución en UML y algún ejemplo de código. Mientras que algunos de los patrones específicos del dominio se pueden representar de esta manera, la mayoría de ellos no se ajustan a esta especificación. Estos patrones son descritos usando términos cercanos a su dominio como se ha mostrado en los ejemplos anteriores, y aunque su campo de aplicación es más específico, estos patrones abordan problemas de un mayor nivel de abstracción ya que sus soluciones reflejan estructuras conceptuales de un dominio de conocimiento y son aplicables a diferentes dominios de aplicación. Finalmente, normalmente en su plantilla no se incluyen ejemplos de código implementando el patrón.

Los patrones específicos del dominio pueden considerarse óptimos puesto que son fruto de la experiencia y tienen definido un alcance más claro que los patrones de diseño de la orientación a objetos, pero no dejan de ser patrones de diseño software ya que describen soluciones de diseño genéricas para sistemas que tienen una aplicación allí donde el software juega un papel clave.

2.1.6. Los patrones como herramienta de comunicación: patrones en ipo

Aunque en el apartado anterior ya se introdujeron los patrones pertenecientes al dominio de la interacción persona-ordenador, esta sección profundizará en ciertos aspectos que son relevantes para los objetivos de este trabajo, en concreto el uso de patrones como herramienta de comunicación, retomando la idea original de Alexander.

El uso de patrones en este dominio se discutió por primera vez en el *workshop* “Lenguaje de patrones para el diseño de interacción” en la conferencia CHI’97, y ha sido un tema de estudio hasta la fecha. Como principal característica diferenciadora, los patrones de interacción persona-ordenador describen propiedades y comportamientos de los sistemas interactivos que pueden ser percibidos por los usuarios, mientras que los patrones de la ingeniería del software generalmente describen la estructura y la ejecución de software, por ejemplo identificando las clases y mensajes entre objetos.

El éxito del diseño de un sistema interactivo requiere la participación de personas procedentes de diferentes disciplinas, como la ingeniería del software, el diseño de interfaces de usuario, el dominio de la aplicación en el cual el sistema será usado, el marketing, la escritura técnica, el diseño gráfico, etc. Teniendo en cuenta este problema, se ha explorado el uso de los patrones como un medio de comunicación. Erickson especula con la posibilidad de una *lingua franca* para todas las personas involucradas en el diseño de un sistema interactivo, incluido los usuarios del sistema final [Erickson, 2000]. Borchers discute el uso de lenguajes de patrones como medio para mejorar la comunicación entre expertos de diferentes dominios [Borchers, 2001]. Teniendo en cuenta estas hipótesis, en [Dearden *et al.*, 2002] se estudia si los lenguajes de patrones pueden mejorar la participación del usuario en el diseño. En dicho estudio se adoptó el mismo proceso participativo que siguió Alexander en [Alexander *et al.*, 1985], recibiendo una respuesta positiva por parte de los usuarios sobre la ayuda que supuso el lenguaje durante la realización de un prototipo, aunque también hubo críticas sobre el formato utilizado para describir los patrones.

Además, los patrones han sido utilizados para enseñar las conceptos básicos de la interacción persona-ordenador a estudiantes de diferentes cursos. En [Borchers, 2002] y [Seffah, 2003], los alumnos concluyeron que los patrones habían resultado útiles tanto para retener los principios de diseño, adoptando rápidamente el vocabulario de los patrones, como para formular sus propias experiencias de diseño.

Finalmente, este área también cuenta con herramientas para la aplicación automática de patrones, en concreto para la generación de interfaces de usuario [Molina, 2003], y se ha empezando a abordar la problemática de organizar y acceder a la gran cantidad de patrones existentes utilizando para ello las tecnología de la web semántica [Henninger *et al.*, 2003]. Un primer paso importante ha sido crear una plantilla de patrón, descrita mediante una DTD llamada PLML (*Pattern Language Markup Language*) [5], que permitirá unificar la apariencia de los diferentes patrones de lenguajes del área.

2.1.7. Formalización de patrones

En la literatura, los patrones aparecen descritos en lenguaje natural siguiendo algún tipo de formato, como se mencionó en la sección 2.1.2, y a veces suelen ir acompañados de dibujos o representaciones gráficas mediante el lenguaje de modelado UML (*Unified Model Language*) [Rumbaugh *et al.*, 1998] y fragmentos de código, que esquematizan la solución

del patrón. Aunque esta representación es útil para entender un patrón y guiar a través de su implementación, carecen de la formalidad necesaria para dar un soporte riguroso al uso de patrones que permita por ejemplo ayudar a un diseñador a identificar un patrón útil para afrontar un determinado problema de diseño.

Muchas son las reticencias en contra de la automatización de los patrones de diseño, como por ejemplo las de Coplien,

> *"... are to be executed by architects with insight, taste, experience, and a sense of aesthetics."*
>
> [Coplien, 1998]

Pero no se puede obviar que el crecimiento en el número de patrones publicados sin un medio efectivo de indexación y la inherente ambigüedad del lenguaje natural utilizado para describirlos dificulta la tarea de transmisión de su conocimiento y de aprendizaje. De hecho, en [Gamma *et al.*, 1994] ya reconocieron que con más de 20 patrones de diseño en el catálogo, es difícil encontrar aquel patrón que dirija un problema de diseño en particular.

Una especificación formal de los patrones permitiría tanto la creación de técnicas de desarrollo basadas en patrones como de herramientas de soporte que podrían ser usadas para construir sistemáticamente soluciones a partir de las especificaciones de patrones (e.g. [Florijn *et al.*, 1997]); verificar la presencia de patrones en los diseños (v.g. [Albin-Amiot y Guéhéneuc, 2001]); incorporar un patrón en el diseño (v.g. [Lucrédio *et al.*, 2003]); o encontrar los patrones adecuados que resuelvan un problema de diseño en un determinado contexto (v.g. [Gomes *et al.*, 2002]).

A continuación se presentan diversas propuestas para la especificación formal de los patrones de diseño agrupadas por el mecanismo base utilizado para su desarrollo: UML o notaciones matemáticas.

Basadas en UML. Actualmente en su versión 1.3, UML propone el uso de colaboraciones como mecanismo para representar la estructura de la solución propuesta en el patrón de diseño en términos de roles y posteriormente aplicar dicha solución en un contexto determinado señalando qué elementos del diseño juegan qué roles de la solución del patrón [Sunyé *et al.*, 2000]. Otras extensiones han sido propuestas para distinguir además de los roles de las operaciones y los atributos [Dong, 2003]. Aunque estas representaciones son útiles para entender la solución del patrón, guiar al diseñador a través de su implementación,

y dejar documentada su utilización, todavía no es suficiente para dar un soporte riguroso al uso de patrones.

En [France *et al.*, 2004] proponen describir las soluciones de los patrones desde dos perspectivas: una vista estructural, descrita mediante un diagrama de clases que describe las clases participantes en la solución del patrón de diseño y sus relaciones; y una vista de interacción, mediante diagramas de secuencia, que describe cómo las instancias de las clases interactúan para llevar a cabo una tarea. Ambas perspectivas se definen mediante la especialización del metamodelo de UML y restricciones OCL parametrizadas que capturan la semántica que no puede ser expresada mediante el metamodelo, como la validación de los elementos que van a participar en la solución. El resultado es un *metamodelo de patrón* que formaliza los aspectos estructurales y dinámicos de la solución de un patrón, a partir del cual se pueden especificar las soluciones de patrones concretos. Esta aproximación permite compartir los patrones de diseño entre herramientas de modelado UML y comprobar de manera inherente la correspondencia en la instanciación del patrón con respecto a su especificación formal.

Basadas en lógica matemática. Otra aproximación todavía más rigurosa es la presentada en los siguientes trabajos, los cuales utilizan como técnica de especificación formal notaciones de lógica matemática.

En [Eden *et al.*, 1997] se describe un lenguaje llamado LePLUS, basado en un fragmento de lógica de orden superior (HOML) donde cada fórmula está formada por una lista de participantes (clases, funciones o jerarquías) y una lista de relaciones entre ellos. Como consecuencia, un patrón es definido como un conjunto de sentencias lógicas. Además, aporta una notación gráfica para su lenguaje permitiendo una mejor legibilidad del patrón especificado. Esta aproximación ha sido implementada en PROLOG permitiendo reconocer y aplicar patrones, definiéndose el patrón como un algoritmo, utilizando pseudo-código, que especifica la secuencia de pasos en su aplicación.

En [Cornils y Hedin, 2000] se modelan los patrones de diseño mediante una gramática con reglas sintácticas sensibles al contexto que especifican los roles y las reglas de los patrones. Los roles son elementos del programa, como clases, métodos y variables que juegan algún papel en el patrón. Las reglas especifican cómo deben combinarse esos elementos. Así, según esta propuesta, los patrones de diseño deben ser primeramente proyectados a roles y reglas. Además, existen dos árboles sintácticos, uno para el programa y otro para los patrones conectados a través de atributos de referencia, que tienen conocimiento sobre la

gramática del lenguaje de programación. Por ahora, la herramienta software que soporta esta formalización permite sólo la aplicación del patrón previa selección del usuario mediante un enlace automático desde los elementos del programa a roles de un determinado patrón de diseño. Este enlace se genera a partir de la información que el usuario proporciona.

Las aproximaciones anteriores se centran en formalizar solamente la parte estructural de la solución del patrón. En [Taibi y Ngo, 2003] se incorpora la especificación del aspecto de comportamiento, es decir el diagrama de secuencia que indica cómo cooperan los participantes descrito en los patrones de diseño de [Gamma *et al.*, 1994]. Por un lado, utilizan lógica de primer orden para especificar el aspecto estructural del patrón (i.e. clases, atributos, métodos, objetos, valores y relaciones) mediante variables, conectores, cuantificadores y predicados. El comportamiento se especifica utilizando lógica temporal de acciones, donde una secuencia de estados (una colección de valores y de relaciones temporales entre los objetos) describe la ejecución de acciones entre los participantes del patrón. Por lo tanto un patrón es modelado como una fórmula de sentencias lógicas, donde un diseño se ajusta a un patrón si y sólo si el diseño satisface dicha fórmula.

Todas estas aproximaciones necesitan del desarrollo de herramientas de soporte propias para el posterior soporte de la utilización de patrones.

2.2. Los sistemas hipermedia

Los sistemas hipermedia, especialmente los basados en la Web, son cada vez más requeridos para proporcionar, acceder, recuperar y manipular información, suponiendo un reto tanto para la creación como para la adaptación de aplicaciones cuyas principales características se expondrán en la sección 2.2.1 y se pueden resumir en: estructuras de navegación cada vez más sofisticadas, comportamientos interactivos más complejos, composiciones multimedia que sean al mismo tiempo usables y estéticas, información personalizada tanto para usuarios como para dispositivos, a la vez que debe preservarse la seguridad de dicha información.

Esta demanda supuso que la mayoría de los desarrolladores dejasen a un lado el diseño conceptual, para directamente acudir a herramientas software que permitiesen una implementación automatizada mediante contenidos y una sencilla puesta en funcionamiento [Fraternali, 1999], anteponiendo los asuntos técnicos a la calidad del producto final, espe-

cialmente en términos del acceso a la información y la usabilidad. Además, el continuo cambio de las tecnologías asociadas a las aplicaciones hipermedia hace de su mantenimiento un proceso muy costoso.

Para dar solución a estos problemas, es ampliamente aceptada la necesidad de aplicar un enfoque de ingeniería del software, es decir, emplear un proceso apropiado para el desarrollo efectivo de aplicaciones de gran escala con una perspectiva larga de vida [Lowe y Hall, 1999]. La sección 2.2.2 esboza el proceso que normalmente se lleva a cabo para acometer el desarrollo de este tipo de aplicaciones, siendo su característica más relevante el estar centrado en el usuario y sus necesidades, lo que conlleva la inclusión de una fase de evaluación que analice la utilidad y usabilidad del sistema imponiendo un modelo iterativo para que los resultados de dicha evaluación puedan ser tenidos en cuenta. Dichas características han impuesto la necesidad de desarrollar métodos propios que se adapten tanto a las peculiaridades del proceso de desarrollo como a las características propias de la hipermedia que hacen a sus aplicaciones diferentes de cualquier otro sistema software. En la sección 2.2.3 se enumeran aquellos métodos más relevantes de la ingeniería hipermedia, como son RMM [Isakowitz *et al.*, 1998], OOHDM [Schwabe y Rossi, 1998], WebML [Ceri *et al.*, 2000], UWE [Hennicker y Koch, 2000], WSDM [De Troyer, 2001], OO-H [Gómez *et al.*, 2001] y ADM [Díaz *et al.*, 2001a].

Sin embargo, aún utilizando los métodos para guiar el proceso de desarrollo de las aplicaciones hipermedia, la productividad y la calidad de estas aplicaciones se podrían mejorar, a la vez que se reduce el coste de desarrollo y de mantenimiento, gracias a la reutilización del diseño, que se tratará en la sección 2.2.4.

2.2.1. Características de las aplicaciones hipermedia

La principal meta de las aplicaciones hipermedia deriva de la idea de Vannevar Bush para emular la memoria humana, proporcionando un acceso efectivo a la información y un modo de administrar el crecimiento del conocimiento [Bush, 1945]. Para ello, la hipermedia se vale de la fusión del hipertexto con la multimedia, cuya sinergia proporciona un paradigma que estructura la información como una red asociativa de *nodos* interconectados mediante *enlaces*.

En el hipertexto, el lector no se enfrenta a una secuencia prescrita como en el caso del libro, sino que dicha estructura anima al lector a explorar la información de manera intui-

tiva, por asociación. Como consecuencia de esta estructura no lineal, la **navegación** es uno de los puntos más críticos en una aplicación hipermedia, ya que a medida que crece la cantidad de información pueden surgir problemas cognitivos. Esto lleva a la desorientación del usuario conocido como el síndrome de "pérdida en el hiperespacio". Para intentar evitar este problema se incorporaron una serie de herramientas de navegación como mapas visuales, índices activos, visitas guiadas, marcas, etc. [Nielsen, 1995].

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 [Nanard y Nanard, 1995].

La hipermedia es una tecnología que está siendo utilizada en diferentes dominios como pueden 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 [Díaz *et al.*, 1999].

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. [Díaz *et al.*, 1999].

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 misma 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 [Kappel *et al.*, 2000].

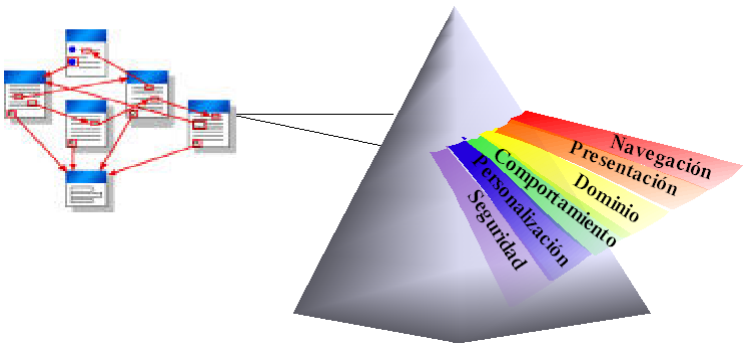


Figura 2.2: Los aspectos de diseño de una aplicación hipermedia

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 [Aedo *et al.*, 2003].

La figura 2.2 muestra un símil entre el fenómeno de la descomposición de la luz y las características de las aplicaciones hipermedia. Al anteponer un prisma a una aplicación hipermedia, se puede ver en el lado derecho que la aplicación queda descompuesta en cada una de las características anteriormente mencionadas (navegación, presentación, estructura del dominio, comportamiento, personalización y seguridad). Sin embargo existe un fuerte acoplamiento entre estas características, que no pueden verse de manera aislada, representado por el lado izquierdo del prisma. Los nodos de información además de tener que ser flexibles con respecto a las necesidades de navegación, tienen que presentar sus contenidos de manera armónica. El dominio de la aplicación determinará el tipo, el número y el modo de navegación de los nodos. El comportamiento puede determinar si su activación en la interfaz da lugar a la navegación o a la ejecución de algún proceso. Finalmente, tanto la personalización como la seguridad están relacionadas con el resto de aspectos ya que presentan diferentes vistas de la misma aplicación hipermedia según los diferentes usuarios de la aplicación.

Se podrían señalar otras características adicionales de las aquí expuestas, dependiendo del tipo de aplicación hipermedia a la que se enfrente el diseñador, como pueden ser colaborativas, distribuidas, comercio electrónico, enseñanza, etc., las cuales quedan fuera del alcance de esta tesis.

De lo aquí expuesto, se desprende que las aplicaciones hipermedia ponen un mayor énfasis en aspectos cognitivos para el acceso y utilización de la información y en aspectos estéticos para la presentación armónica de dicha información [Nanard y Nanard, 1995] que en el procesamiento de los datos y, por ello, todas las características aquí expuestas deberán ser tratadas como aspectos de diseño en el proceso de modelado de las aplicaciones hipermedia por métodos especializados de ingeniería.

2.2.2. Proceso de desarrollo

Para llevar a cabo el desarrollo de una aplicación de tales características de una manera más eficiente y efectiva, es necesaria la consecución de una serie de actividades que conformen un proceso de desarrollo. Además, hay ciertas peculiaridades en el proceso de desarrollo de estas aplicaciones que requieren de enfoques distintos a los que se aplican en otras disciplinas.

Las aplicaciones hipermedia tienen como objetivo conseguir que el usuario pueda llevar a cabo algún tipo de tarea, como comprar o aprender algo. El conocimiento sobre cómo realizar dicha tarea reside en el usuario y es la aplicación la que debe responder de la manera que éste espera. Esto conlleva un enfoque de desarrollo **centrado en el usuario** y en sus necesidades, que promueva la participación activa de éste con el objetivo de construir un producto usable [Gould *et al.*, 1997]. La otra peculiaridad está relacionada con las personas involucradas en el desarrollo de la aplicación. Como ya se ha mencionado anteriormente, por un lado, un miembro importante del equipo de desarrollo es el usuario final, pero además el desarrollo de una aplicación hipermedia involucra aspectos tecnológicos, resueltos por ingenieros software y programadores, y también aspectos cognitivos y creativos resueltos por especialistas en diseño, lingüistas, autores y diseñadores multimedia. Se trata pues de un equipo multidisciplinar con miembros procedentes de diferentes áreas de conocimiento y con diferentes habilidades [Deshpande y Hansen, 1998]. Este hecho implica que tanto el proceso de desarrollo como los métodos de diseño deberán contar con actividades y productos que fomenten la cooperación y sinergia entre dichos miembros.

Dentro del proceso de desarrollo hipermedia se pueden destacar tres fases fundamentales: el análisis, el diseño y la evaluación.

El **análisis** es la fase que tiene como objetivo estudiar y documentar las características o requisitos que la aplicación a desarrollar debe tener, con objeto de convertirlos en requi-

sitos estables que puedan emplearse para verificar la usabilidad y utilidad del sistema final. Los requisitos pueden hacer referencia a distintas facetas del producto, pudiéndose clasificar en funcionales (funciones que el sistema debe realizar), no funcionales (restricciones a los requisitos funcionales relacionados con la eficiencia, consistencia, reusabilidad, etc.) [IEE, 1990], y de usabilidad (nivel aceptable de utilización y de aceptación del producto final por parte del usuario) [Preece *et al.*, 2002].

En la fase de **diseño**, la especificación de la aplicación se convierte en una descripción de cómo la aplicación cumplirá con las especificaciones definidas, es decir los requisitos son transformados en soluciones lógicas aplicando una abstracción de alto nivel que describa la estática y la dinámica del sistema, pero sin basarse en asuntos técnicos. A partir de las características expuestas en la sección anterior, las cuales describen como son las aplicaciones hipermedia, esta fase debería involucrar diferentes perspectivas complementarias que describan la aplicación en términos de navegación, contenido, comportamiento, estructura, presentación, y seguridad. El uso de un **modelo de diseño** proporciona el marco formal para apoyar la transformación de las necesidades del usuario en productos de diseño relacionados con estas perspectivas.

Finalmente, mediante la **evaluación** de prototipos generados a partir del diseño, se pueden visualizar los diferentes elementos y funcionalidades de la aplicación, ayudando así a diseñadores y usuarios a mejorar la usabilidad del sistema.

En la figura 2.3 se representa una secuencia para las actividades anteriormente expuestas, denominado **modelo de proceso iterativo** muy utilizado en el proceso de desarrollo hipermedia [Lowe y Hall, 1999] ya que permite incorporar al usuario tanto en la fase de análisis como en la evaluación de los prototipos. Los resultados de la evaluación pueden hacer replantear cualquiera de las fases previas, como señalan las flechas de la figura.

Finalmente, para llevar a cabo las fases propuestas en el proceso de desarrollo, se precisan de métodos específicos que guíen al diseñador a través de una serie de pasos y actividades que permitan separar los diferentes niveles de abstracción de aspectos de diseño a tratar durante el desarrollo. Sin embargo, la utilización de estos métodos requiere diseñadores y desarrolladores hábiles e incluso con una formación en ingeniería del software. Aunque a veces, este requisito se ve aliviado por **herramientas software** que soportan al método de diseño y proporcionan el desarrollo automatizado de prototipos.

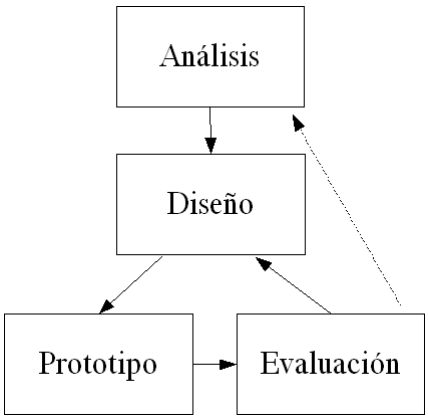


Figura 2.3: Modelo de proceso iterativo [Lowe y Hall, 1999]

2.2.3. Métodos de desarrollo

Los diferentes aspectos que abarca el diseño de las aplicaciones hipermedia y los requisitos de su proceso de desarrollo involucran decisiones no triviales que inevitablemente influyen en el producto final. Por ello, es necesario contar con métodos y técnicas de modelado que guíen al diseñador hipermedia a través de las diferentes fases de desarrollo a la vez que le proporcionan los mecanismos necesarios para afrontar el complejo modelado de estas aplicaciones.

A continuación se presentan una serie de métodos de diseño hipermedia, los cuales han sido seleccionados tanto por su relevancia en el área como por sus significativas aportaciones al proceso de desarrollo o al modelado hipermedia.

- *RMM* (*Relationship Management Methodology*). RMM [Isakowitz *et al.*, 1998], es por ahora es el único método hipermedia que propone un ciclo de desarrollo software completo, desde el análisis de viabilidad hasta la evaluación, además de soportar una aproximación de modelado tanto *ascendente* como *descendente*. El método se basa en el modelo RMDM (Relationship Management Data Model), inspirado en el modelo Entidad-Interrelación y HDM. El método comienza representando la estructura de información, para añadir posteriormente, atributos de entidades, enlaces y anclas. Cuenta con una herramienta de apoyo al modelado llamada RMMCase [29]. Para evaluar la aplicación sugieren utilizar la técnica propuesta de HDM.

- *OOHDM* (*Object-Oriented Hypermedia Design Method*). OOHDM [Schwabe y Rossi, 1998] acomete el modelado de una aplicación desde cuatro actividades siguiendo un proceso iterativo e incremental. El punto de partida es el diseño conceptual, basado en la técnica de modelado OMT. A continuación el diseño de la navegación permite construir diferentes vistas del mismo modelado conceptual. La siguiente actividad es el diseño abstracto de la interfaz, donde los contenidos son organizados y diferentes interfaces pueden ser definidas para el mismo modelo de navegación, permitiendo así la personalización de la aplicación a partir de la construcción de diferentes vistas para cada perfil de usuario. Un punto importante es la inclusión de patrones de diseño en el proceso de desarrollo. En la última actividad, la de implementación, proponen una serie de pasos para pasar el diseño a un entorno de implementación.

- *WSDM* (*Web Site Design Method*). WSDM [De Troyer, 2001] es un método centrado en el usuario que consta de las siguientes fases: Modelado del Usuario, Diseño Conceptual, Diseño de Implementación e Implementación. En la primera fase, se identifican los usuarios del sitio web y se clasifican en tipos de usuario, de acuerdo a sus requisitos de información. Un tipo de usuario puede ser dividido en perspectivas y el sistema puede modelarse desde el punto de vista de cada tipo de usuario. En la fase Diseño Conceptual, los requisitos de información de cada tipo de usuario se describen mediante un modelo de objetos. Además, en esta fase se construye el modelo navegacional desde cada perspectiva. En la fase de Diseño de la Implementación se diseña la apariencia del sitio web.

- *WebML* (*Web Modeling Language*). WebML [Ceri *et al.*, 2000] especifica un proceso de diseño iterativo que guía el proceso de desarrollo desde la recogida de requerimientos hasta la personalización del diseño. Primero se expresan los contenidos y datos del sitio en el modelo estructural. Después, el modelo hipertextual define las páginas y contenidos que componen el hipertexto y la forma en la que son enlazadas. Además, los contenidos pueden organizarse en regiones autocontenidas de la pantalla, las cuales pueden mostrarse simultáneamente o no. Entonces se especifican las capas de las páginas en el modelo de presentación. Además, WebML incluye la noción de grupo (conjunto de usuarios) y usuario (individual) que son modelados como una entidad especial en el modelo estructural permitiendo la personalización del hiperdocumento. El comportamiento interactivo se soporta mediante las unidades de operación que son

enlazadas con otras unidades y pueden ser predefinidas o construidas por el diseñador. Todo el proceso de diseño está soportado por un entorno CASE llamado WebRatio [1].

- *OO-H* (*Object-Oriented-Hypermedia*). OO-H [Gómez *et al.*, 2001] comienza con la estructura del dominio de la información, desde los puntos de vista estático y dinámico, recogido en un diagrama de clases utilizando UML. Este diagrama se extiende con el diagrama de acceso navegacional, que define una vista navegacional y el diagrama de presentación abstracta que recoge los conceptos relacionados con la presentación. Estas últimas vistas están soportadas por un catálogo de patrones que ayudan al diseñador a obtener la interfaz deseada. Los usuarios pueden ser modelados dentro del diagrama de clases y asociarles un conjunto de reglas de personalización. Además cuenta con una herramienta CASE, llamada VisualWade [2], que soporta todo los aspectos del método y permite generar aplicaciones web.

- *ADM* (*Ariadne Development Method*). ADM [Díaz *et al.*, 2001a, Díaz *et al.*, To be published], basado en el modelo de referencia Labyrinth [Díaz *et al.*, 2001b] propone un proceso sistemático, integrado e independiente de plataforma basado en tres fases: Diseño Conceptual, Diseño Detallado y Evaluación. Además, un conjunto de reglas de validación y verificación ayudan al usuario a mantener la integridad y completitud del diseño acometido con respecto a la semántica del método. Durante el Diseño Conceptual se llevan a cabo las siguientes actividades sin imponer ningún orden de realización: definición de la estructura lógica del dominio de la aplicación mediante nodos y mecanismos de abstracción; estudio de las funciones del sistema tanto navegacionales como otros servicios; especificación de los contenidos tanto en su dimensión espacial como temporal, anclas, enlaces, atributos y eventos asociados a los nodos; identificación de los diferentes tipos de usuarios mediante roles y grupos que hacen posible definir presentaciones dependientes del usuario; y la definición de políticas de acceso que deciden qué acciones son permitidas siguiendo un modelo RBAC. Además cuenta con una herramienta de soporte al modelado y generación de prototipos llamada AriadneTool [11].

A modo de resumen, la tabla 2.1 muestra cómo los métodos de diseño aquí presentados dan soporte o no a las características marcadas en negrita en las secciones 2.2.1 y 2.2.2. Dicha tabla muestra en sus columnas las características más relevantes del proceso de

desarrollo hipermedia, señalando para cada método si lo recoge de manera completa, parcial o no. Para una comparativa exhaustiva entre los diferentes métodos se puede consultar [Montero *et al.*, 2002, Montero *et al.*, 2003b].

2.2.4. Problemas en el desarrollo

Aún utilizando los métodos anteriormente expuestos para guiar el proceso de desarrollo de las aplicaciones hipermedia, 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 [Barry y Lang, 2001] 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 (p.e. desorientación, sobrecarga de información, pobre usabilidad, insatisfacción o 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 (técnicos, de seguridad, de interacción, etc.) para los cuales a veces puede no poseer las habilidades necesarias [Bolchini, 2000].

Por otro lado, conocidos son los problemas de comunicación existentes con los usuarios durante la captura de requisitos y la fase de análisis en la ingeniería del software [13]. Este problema puede verse agravado si el desarrollo es llevado a cabo por un equipo de trabajo multidisciplinar y los usuarios son involucrados durante todo el proceso de desarrollo, como es el caso de la ingeniería hipermedia. Sus integrantes son 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 marketing, y, por supuesto, los usuarios finales, teniendo que trabajar todos ellos conjuntamente hacia la misma dirección: cumplir las expectativas de los usuarios [Preece *et al.*, 2002].

Para afrontar el primero de los asuntos, la reutilización en el diseño hipermedia es crucial pues permite mejorar la productividad y la calidad de las aplicaciones, además de reducir el coste tanto en el desarrollo como en el mantenimiento. La reutilización consiste en aprovechar cualquier esfuerzo realizado en un desarrollo previo para reducir el esfuerzo futuro [Nanard *et al.*, 1998]. Para ello, la reutilización puede involucrar aspectos como datos, componentes software y experiencias de diseño. La reutilización de datos pasaría por compartir los contenidos multimedia (textos, imágenes, vídeos)

Métodos	Centrado en el usuario	Iterativo	Análisis	Diseño	Evaluación	Modelo formal	Herramienta sw
RMM	C	C	P	C	C	RMDM	RMCase
OOHDM	C	C	C	C	N	basado en ODMG	-
WSDM	C	C	N	C	N	basado en UML	-
WebML	C	C	C	C	C	basado en E-R	WebRatio
OO-H	C	C	N	C	N	basado en UML	VisualWade
ADM	C	C	N	C	C	Labyrinth	AriadneTool

C = completamente soportado; **P =** parcialmente soportado; **N =** no soportado;

Tabla 2.1: Análisis de los métodos con respecto al proceso de diseño

[Garzotto *et al.*, 1997]. Los componentes software pueden ser reutilizados a nivel de componentes [Gaedke y Rehse, 2000] o actualmente a nivel de servicios [7]. Sin embargo, el tipo más importante de reutilización para los diseñadores es la experiencia de otros diseñadores, que permite aprovechar las ideas más que las implementaciones. El ahorro en costes y tiempo que se obtiene en la reutilización del software se logra en mayor medida en la reutilización del conocimiento, debido al enorme esfuerzo que conllevan los procesos de adquisición de conocimiento de un dominio, la construcción de su modelo conceptual, y la formalización e implementación de tales conocimientos [Poulin, 1996]. En la ingeniería del software, esta reutilización del conocimiento es identificada mediante **patrones de diseño**, como se ha visto en la sección 2.1.2. Los patrones de diseño 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 en un formato simple y comprensible tanto para expertos como para noveles, sino que posibilitan la reutilización de la misma experiencia una y otra vez en diferentes aplicaciones.

Con respecto al segundo problema, el área hipermedia cuenta con modelos como el propuesto en [Bolchini *et al.*, 2003] para documentar y especificar formalmente los requisitos capturados durante la actividad de análisis, pero desde el punto de vista del analista. Sin embargo no se ha contemplado cómo mejorar la comunicación entre los usuarios y el equipo de desarrollo siguiendo un diseño participativo. En la sección 2.1.6 se ha expuesto como los lenguajes de patrones se han utilizado en el dominio de la interacción persona-ordenador como mecanismo de comunicación, por un lado, entre usuarios y diseñadores a la vez que definían el prototipo inicial de un sistema interactivo, y por otro, para introducir a alumnos en los conceptos básicos del área.

En conclusión, puede decirse que los patrones de diseño pueden constituir una solución de ambos problemas siempre que puedan integrarse de forma eficiente en los métodos de diseño y que se presenten de manera que favorezcan la comunicación.

A continuación, se muestra cómo los patrones de diseño han sido adoptados por la comunidad hipermedia y cómo son utilizados.

2.3. Patrones de diseño en la ingeniería hipermedia

Como se ha mostrado en la primera parte de este capítulo, 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 permiten aplicar el conocimiento de expertos del dominio a la resolución de problemas recurrentes. Por tanto, no es sorprendente que la comunidad hipermedia, como el resto de comunidades ya presentadas, adoptase también el concepto de patrón para beneficiarse de la efectividad de su aplicación en el desarrollo de sus propios sistemas. Los patrones de diseño hipermedia están pensados para capturar soluciones a problemas de diseño derivados del modelado de las características expuestas en la sección 2.2.1 (navegación, presentación, dominio de la aplicación, comportamientos interactivos, personalización y seguridad), es decir, por ejemplo cuando un diseñador menos experimentado tenga que resolver un problema específico de navegación, sería útil proporcionarle la solución dada para una situación análoga por un diseñador más experimentado. Con respecto a los métodos de diseño, como los enumerados en la sección 2.2.3, los patrones de diseño son capaces de trascender las restricciones de una notación de diseño predefinida porque ayudan a capturar la esencia, los aspectos universales de los problemas de diseño y sus soluciones.

En 1996, en el seno del grupo de investigación del LIFIA liderado por Gustavo Rossi et al. surgen los primeros trabajos sobre patrones de diseño aplicados al desarrollo de aplicaciones hipermedia, dándose a conocer algunos patrones para la creación de aplicaciones orientadas a objetos con funcionalidad hipermedia en [Rossi *et al.*, 1996] muy cercanos al estilo de los patrones de [Gamma *et al.*, 1994]. Sus siguientes trabajos se centran ya en problemas de diseño de las aplicaciones hipermedia [Rossi *et al.*, 1997] y se hace una primera aproximación para la creación de un lenguaje de patrones aplicado a este dominio [Garrido *et al.*, 1997]. Además, en [Rossi *et al.*, 1997] se puso de manifiesto la necesidad de motivar a la comunidad hipertexto a discutir el asunto y producir un catálogo de patrones relacionados. Este periodo termina con un *workshop* [8] donde se discuten las principales carencias con las que contaba el espacio de patrones de diseño hipermedia y se proponen una serie de intenciones futuras. Además en ese mismo año se crea el primer repositorio web de patrones hipermedia [12] y se investiga cómo integrar los patrones en el proceso de diseño [Rossi *et al.*, 1999, Rossi y Lyardet, 1999]. En [German y Cowan, 2000] se hace un último esfuerzo por recopilar todos los patrones existentes e intentar contar con un catálogo

de patrones común, pero más que un catálogo el resultado final fueron una serie problemas que hacían de la tarea casi algo imposible.

Sin embargo con el auge de las aplicaciones web y la necesidad de que el desarrollo esté centrado en el usuario final, reaparecen los patrones de diseño pero con una nueva perspectiva. Aunque tanto los patrones anteriores como éstos tienen en común las necesidades de los usuarios, la diferencia radica en cómo son explicadas las soluciones que proponen los patrones. Mientras que en el primer caso se utiliza un vocabulario propio de los métodos de diseño, que los hace comprensibles sólo a diseñadores hipermedia experimentados, este nuevo estilo de patrones, que se han recogido tanto en libros [van Duyne *et al.*, 2002, Graham, 2003b] como en repositorios web [33, 17], están descritos con un lenguaje comprensible tanto para diseñadores como para usuarios, permitiendo nuevos modos de uso de los patrones.

Todos estos asuntos serán tratados en profundidad en los siguientes apartados.

2.3.1. Características de los patrones de diseño hipermedia

El campo de la hipermedia es muy extenso, incluyendo áreas como la hipermedia adaptativa, la computación ubicua, la hipermedia educativa o la web, las cuales cuentan todas ellas con sus propios patrones de diseño [Koch y Rossi, 2000, Landay y Borriello, 2003, Hubscher y Frizell, 2002, Bonura *et al.*, 2002], respectivamente. Además pueden encontrarse para diferentes niveles de granularidad, como a nivel de arquitectura [Noack *et al.*, 2000] o a nivel de programación [Weiss, 2003]. Como ya se ha mencionado con anterioridad, el contexto en el que se centra esta tesis son los patrones que resuelvan problemas relacionados con el diseño genérico de cualquier aplicación hipermedia[1], por lo que estos últimos tipos de patrones (de arquitectura y de programación) quedan fuera del alcance del estudio que a continuación se presenta.

Los patrones de diseño hipermedia formarían parte de los patrones denominados *específicos del dominio*, ya presentados en la sección 2.1.5, y su campo de conocimiento es la resolución de problemas recurrentes en el diseño de las aplicaciones hipermedia. Por lo

[1] El término **Patrones de diseño hipermedia** se utilizará para referirse a aquellos patrones que resuelvan problemas relacionados con aspectos de diseño, mientras que el término **Patrones hipermedia** hará referencia a aquellos patrones relacionados con cualquier tipo de aplicación hipermedia, independientemente de su granularidad

tanto, comparten las características señaladas en dicha sección, adaptadas al campo de la hipermedia, y otras propias de su dominio.

Al igual que los patrones de interacción persona-ordenador, la principal diferencia con los patrones de la comunidad de orientación a objetos es el vocabulario utilizado para describir los patrones. Los patrones de diseño hipermedia describen problemas relacionados con diferentes aspectos de diseño como la navegación, la presentación de la interfaz, utilizando términos propios del área hipermedia y de las interfaces de usuario, para proporcionar soluciones de un nivel más alto de abstracción, sin llegar a describir a aspectos de ejecución software. Por ello, la mayoría de los patrones siguen un formato más simplificado con respecto a los patrones software tradicionales, dejando a un lado los campos relacionados con la implementación, para dar al diseñador más libertad durante la implementación.

Es necesario aclarar que un patrón de diseño debería usarse sólo cuando su aplicación sea realmente ventajosa. 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. Los elementos que al menos deberían describir un patrón de diseño hipermedia son [Bolchini, 2000]:

- Nombre, un único identificador para el patrón de diseño hipermedia.
- Exposición del problema.
- Solución o conjunto de posibles soluciones.
- Discusión de las ventajas y desventajas de cada una de las soluciones. Una discusión exhaustiva incluye la explicación de las restricciones que los diseñadores deben tener en cuenta.
- Ejemplos de aplicación. El contexto resultante puede comunicarse de manera sencilla mostrando cómo trabaja el patrón en aplicaciones hipermedia existentes.
- Identificación de los patrones relacionados: no sólo con el nombre del patrón sino señalando el tipo de relación que tienen (un patrón puede ser una especialización o generalización de otro, dos o más patrones pueden también ser utilizados de manera conjunta o combinada).

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. Esto implica que los patrones tienen que ser accesibles a personas con diferentes habilidades y formación, lo cual puede dificultar el reconocimiento de las situaciones en las cuales los patrones de diseño se pueden reutilizar y cómo implementarlos con estructuras concretas que se ajusten a ellos.

Para finalizar este apartado, señalar una peculiaridad de los patrones de diseño hipermedia. Debido a que los aspectos de diseño involucrados en el diseño de los sistemas hipermedia están estrechamente relacionados entre sí, como se describió en la la sección 2.2.1, esta misma característica se ve proyectada en la gran mayoría de los patrones de diseño por lo que es normal encontrar patrones que plantean el mismo problema pero desde aspectos de diseño diferentes.

2.3.2. Clasificación de patrones de diseño hipermedia

El principal problema que puede encontrar un diseñador a la hora de afrontar la utilización de patrones es identificar el problema correctamente, es decir, ser capaz de dividir un problema complejo, como es el desarrollo de una aplicación hipermedia, en problemas más pequeños. Por lo tanto, una vez que el dominio cuenta con un número considerable de patrones es necesario disponer de catálogos completos que agilicen su aplicación durante todo el proceso y ayuden a una recuperación sencilla y útil. A continuación se presentan las diferentes clasificaciones de patrones hipermedia existentes.

- Rossi et al. realizaron la primera clasificación [Rossi *et al.*, 1997] en la que los patrones son organizados dentro de dos categorías, denominadas *sistemas hipermedia* y *aplicaciones hipermedia*, esta última es subdividida en diferentes aspectos de diseño. Aunque inicialmente sólo se incluían el diseño navegacional y el de presentación. Lyardet realizó un refinamiento en el que se distinguía la organización de la información, tanto estructural como navegacional, organización de la interfaz e implementación [Lyardet *et al.*, 1998].

- H. Nanard y J. Nanard presentan un espacio multidimensional compuesto de los siguientes ejes: *aplicación hipermedia*, que recoge los aspectos de diseño de una aplicación como el tipo de aplicación, el modelado del dominio, la retórica, la estructura,

la dinámica, la presentación abstracta y la interfaz concreta; *diseño y desarrollo*, que hacen referencia al ciclo de vida de la aplicación hipermedia como modelos y métodos de diseño, gestión de procesos, diseño, evaluación, implementación; *sistema hipermedia* como producto final teniendo en cuenta la arquitectura, comportamiento y producción; *factores humanos* recoge criterios de ergonomía, culturales, de cooperación, perfil de usuario, etc. [Nanard y Nanard, 1999]. La idea es proyectar cada patrón sobre este espacio multidimensional y analizar las relaciones entre los elementos del patrón y los subespacios reflejados.

- Garzotto et al. proponen una clasificación basada en dos criterios [Garzotto *et al.*, 1999]: el *alcance del diseño*, es decir el problema de diseño que trata el patrón, incluyendo la estructura, navegación y presentación/interacción de la aplicación; y el *nivel de abstracción*, donde cada patrón debería estar descrito en tres diferentes niveles de abstracción (requerimientos de usuario, diseño, detallado) para que el diseño pueda ser acometido desde diferentes niveles [Paolini y Garzotto, 1999].

- Germán y Cowan proponen una única clasificación a nivel de diseño: arquitectural, construcción de componentes, presentación, comportamiento e interacción de usuarios, la cual queda abierta a otros tipos de patrones que sean descubiertos [German y Cowan, 2000].

- van Duyne et al. han organizado los patrones en grupos, comenzando con una clasificación de sitios webs y profundizando progresivamente en elementos concretos de la aplicación. Cada uno de estos grupo de patrones contiene una colección temática de patrones relacionados [van Duyne *et al.*, 2002].

Es necesario señalar que difícilmente un patrón puede quedar clasificado por un único criterio porque el diseño de las aplicaciones hipermedia es un proceso integral que considera simultáneamente diferentes aspectos, y por lo tanto se ven plasmados en sus soluciones. Sin embargo, el tener varias clasificaciones para el mismo espacio de patrones puede aportar los siguientes beneficios [Bolchini, 2000]:

- Mejora el entendimiento de las relaciones entre los patrones de diseño. Agrupar diferentes patrones bajo el mismo criterio permite a los diseñadores compararlos e identificar similitudes no explícitas entre ellos.

- Mejora el aislamiento del estado del problema. Sin un correcto aislamiento del problema no se puede usar un patrón de manera efectiva. Las clasificaciones podrían facilitar la comparación entre problemas de diseño similares y la identificación de un problema específico entre un conjunto de problemas próximos en un único aspecto de diseño.

- Estudiar el mismo patrón de diseño desde diferentes perspectivas. Como ya se ha mencionado, un mismo patrón de diseño puede involucrar diferentes aspectos de diseño y dependiendo del autor, la solución de diseño pondrá más énfasis en una perspectiva que en otra.

Resumiendo, y atendiendo sólo a los criterios referentes a las actividades de diseño para las cuales se han categorizado patrones, se suelen distinguir los siguientes tipos de patrones [Garzotto *et al.*, 1999, Lyardet *et al.*, 1999, German y Cowan, 2000]:

Estructurales: son aquellos que están relacionados con la especificación de tipos de información y cómo están organizados dichos tipos. Un ejemplo de patrón estructural es el llamado *Ciclo* donde el lector vuelve a una página anteriormente visitada para tomar un nuevo camino [Bernstein, 1998]. Los ciclos aparecen de manera natural en el hipertexto.

Navegacionales: están relacionados con la especificación de caminos de navegación a través de los elementos de información. Un ejemplo de patrón navegacional es el llamado *Index Navigation* cuyo objetivo es proporcionar acceso rápido a un grupo de nodos mediante la creación de un conjunto de enlaces que acceden a cada uno de esos nodos [Garzotto *et al.*, 1999].

Presentación: están relacionados con la organización de la comunicación multimedia y de la interacción con el usuario. Por ejemplo el patrón *Información por demanda* se preocupa de cómo organizar la interfaz de tal manera que sea el usuario el que decida cuánta información desea ver en la pantalla, por ejemplo mediante una serie de botones que muestre u oculte información [Rossi *et al.*, 2000a].

Los más utilizados son los patrones de diseño relacionados con la navegación, debido a que es una característica intrínseca y diferenciadora de la tecnología hipermedia.

El siguiente paso, una vez que se tiene un espacio en el cual los patrones pueden ser catalogados, es hacer explícitas las relaciones subyacentes entre dichos patrones. Un patrón, rara vez se aplica de manera aislada, sino que su aplicación puede conllevar a la utilización o no de otros patrones. En la siguiente sección se describe en qué contexto son aplicados los patrones de diseño hipermedia y cuáles son los principales motivos de su poca aceptación a pesar de que su utilización permite mejorar la calidad del diseño y reducir los tiempos y costes de implementación, debido a la reutilización de la experiencia de diseñadores y desarrolladores del área.

2.3.3. Aplicación de los patrones de diseño

Como ya se ha mencionado reiteradamente, los patrones de diseño facilitan la reutilización de estructuras a un nivel alto de abstracción, ya que proporcionan soluciones a problemas que se repiten de manera recurrente e incluso en diferentes contextos. Por lo tanto, los diseñadores necesitan de técnicas que permitan capturar la especificación de las estructuras que registran los patrones y reproducirlas fácilmente durante el proceso de diseño de las aplicaciones hipermedia.

En [Nanard *et al.*, 1998] proponen por un lado la utilización de *Golden rules* que ayuden a decidir dónde reutilizar los patrones de diseño y, por otro, *Constructive Templates* que ayuden a capturar las abstracciones de diseño para la implementación de los patrones. La primera técnica . La segunda técnica actúa como puente para la reutilización desde el diseño a la implementación, especificando estructuras y componentes reutilizables usados durante el proceso de diseño. Estas técnicas utilizan como elementos de representación los propuestos por HDM [Garzotto *et al.*, 1993].

En [Garzotto *et al.*, 1999] proponen su utilización dentro del marco de los métodos de diseño como una técnica complementaria, bien para solventar el hueco existente entre el modelado conceptual y la implementación del sistema, debido a que los patrones permiten hacer una correspondencia entre el problema y la solución , o bien, para enriquecer dichos métodos con primitivas de más alto nivel de abstracción, porque los patrones permiten expresar conocimiento con una única notación [Lyardet *et al.*, 1999],

Finalmente, en [Bolchini, 2000] se señala que el alcance de los patrones de diseño hipermedia es proporcionar estrategias efectivas que rellenen el hueco entre los requerimientos y el diseño.

De los métodos aquí expuestos, por un lado se encuentra OOHDM [Rossi y Lyardet, 1999], como el único que incluye los patrones dentro de su proceso de modelado, mediante la descripción y clasificación de los patrones en términos de sus propios elementos, pero no proporciona un marco común a los métodos de diseño. Por otro lado, se encuentra OO-H [Gómez *et al.*, 2001] que los incluye de manera integrada en su herramienta CASE. En cuanto al resto, los posibles motivos de su poca aceptación se pueden deber a la falta de un vocabulario común que los describa, como puede verse en el estudio realizado en [German y Cowan, 2000] donde se recopilan todos los patrones publicados hasta la fecha y se muestran como diferentes patrones describen el mismo problema, pero desde diferentes vistas, lo que dificulta aún más la creación de una clasificación y un sistema de patrones común, que guíe al diseñador en la tarea de determinar cuál utilizar según en que fase o producto se encuentre, independientemente del método de diseño utilizado.

2.4. Resumen del capítulo

Partiendo del eje central de este trabajo, los patrones hipermedia, este capítulo ha revisado los contenidos y el estado de la cuestión de las materias que conforman el marco teórico.

Se comenzó por el origen de los patrones en la arquitectura urbanística, para después hacer una revisión de los patrones de diseño en la ingeniería del software y cómo su campo de aplicación se amplió a otras áreas, destacando el área de la interacción persona-ordenador por su estrecha relación con los sistemas hipermedia. De cada una de ellas se puede destacar:

De la arquitectura urbanística: Alexander concibe la idea de patrón como un mecanismo para registrar un buen diseño que se manifiesta a través de diferentes sitios y culturas, mediante una regla contexto-problema-solución, a la vez que proporcionaba un vocabulario común entre usuarios y arquitectos.

De la ingeniería del software: Los patrones fueron adoptados como mecanismo de reutilización software a diferentes niveles (arquitectura, diseño, programación). De su uso continuo se concluyó que los patrones proporcionan un vocabulario común entre ingenieros software, mejoran la documentación de los sistemas desarrollados, a la vez que ayudan a reestructurar el sistema. Además, ha habido varios intentos de dotar

de una formalización a los patrones para el desarrollo de herramientas software de soporte.

De la interacción persona-ordenador: Vuelven a la idea original de Alexander, los patrones como una *lingua franca* accesible para todos. Para ello han realizado diversos estudios empíricos donde los patrones son utilizados como herramienta de comunicación entre el grupo de desarrollo, incluidos los usuarios, durante la elaboración de prototipos iniciales; y como herramienta docente para transmitir los conceptos básicos del dominio en cursos de iniciación a los sistemas interactivos.

Antes de abordar el caso de los patrones de diseño hipermedia, se quiso poner de manifiesto que las aplicaciones hipermedia poseen una serie de características, que las hace diferentes del resto de aplicaciones software, como por ejemplo sofisticadas estructuras de navegación, comportamientos interactivos complejos, composiciones multimedia, y el acceso a la información de manera personalizada y segura; y que la existencia de un conjunto de métodos formales para guiar al diseñador en los anteriores aspectos de diseño mediante un proceso sistemático, iterativo y centrado en el usuario, hacen de esta disciplina una ingeniería. Sin embargo dos problemas se hacen evidentes:

1: La complejidad de las aplicaciones hipermedia, el breve periodo de tiempo impuesto por el mercado para el desarrollo de estas aplicaciones unido al vertiginoso cambio de las tecnologías, y la posibilidad de que un diseñador, tanto si es experto como no, puede llegar a adoptar una solución inadecuada a un determinado problema, imponen la necesidad de una reutilización software a nivel de diseño más que a nivel de implementación.

2: Los integrantes del grupo de desarrollo incluyen 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 marketing, y, por supuesto, los usuarios finales, teniendo que trabajar todos ellos conjuntamente, lo que impone la necesidad de contar con herramientas que asistan a los usuarios en la búsqueda, selección y aplicación de un patrón.

La evidencia de que los mismos motivos que impulsaron la adopción de los patrones de diseño en las áreas de la ingeniería del software y de la interacción persona-ordenador

podrían encontrarse en el área hipermedia como solución a estos problemas, llevó a la realización de un estudio del estado actual de los patrones de diseño hipermedia, que se puede resumir en los siguientes puntos:

- Existen dos estilos de patrones que se corresponden con las perspectivas de la ingeniería del software y de la interacción persona-ordenador.
- Existen diversos criterios de clasificación aunque no se han utilizado como base para elaborar un catálogo común, y menos aún se ha definido un lenguaje de patrones.
- La aplicación de los patrones está estrechamente ligada a los métodos de diseño y algunas herramientas CASE los soportan.
- No ha habido ninguna tentativa de formalización para el desarrollo de herramientas de soporte para la organización, recuperación o exploración de los patrones, salvo algunos repositorios web.

Capítulo 3

Planteamiento del problema

A pesar de las bondades evidentes que aportan los patrones de diseño, el estudio realizado sobre los patrones de diseño hipermedia muestra que no han tenido la misma repercusión y evolución que en el área de la ingeniería del software o de la interacción persona-ordenador, no siendo una práctica habitual en el proceso de diseño de las aplicaciones hipermedia. Por ello, a continuación se analizan las principales causas y carencias extraídas de dicho estudio (véanse las secciones 3.1 y 3.2) que han llevado a abordar la realización de esta tesis, presentando además una serie de requisitos (véase la sección 3.3) que complementan a los objetivos definidos en la sección 1.3 del capítulo 1 "Introducción".

3.1. Problemática del 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 o en alguno de los repositorios web disponibles. Entre todos los patrones, 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 des-

criben el mismo problema pero desde diferentes puntos de vista [German y Cowan, 2000], 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 se quiere aplicar**. En este caso los patrones de diseño hipermedia serán aplicados durante el proceso de diseño basado en métodos que, como se ha puesto de manifiesto en la sección 2.2.3 "Métodos de desarrollo", no comparten el mismo modelo de referencia. En la actualidad, no existe **un mecanismo detallado** que permita dicha adaptación. Además, 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 (p.e. autor, diseñador, artista o ingeniero). Así, la transferencia de conocimiento entre el patrón y el diseñador se ve comprometida al estar obligado éste a reinterpretar y reescribir el patrón en términos del método que él 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, ya que no existe **un lenguaje de patrones** que guíe al diseñador en la selección del siguiente patrón según el problema de diseño del que partió.

El procedimiento anteriormente descrito, evidencia que el uso de los patrones no es un proceso trivial y que son necesarias una serie de herramientas que asistan a los diseñadores. Además, el diseñador o el desarrollador necesita un alto conocimiento en el diseño de aplicaciones y un alto nivel de abstracción para reconocer los problemas y sus potenciales soluciones, así como de la existencia y manejo de los patrones de diseño. Quizás en la comunidad de la orientación a objetos esta cuestión no sea un asunto trascendental, ya que todos los participantes en el desarrollo de aplicaciones son profesionales del software, y la adquisición de estas habilidades se produce de manera natural. Sin embargo, esta situación es diferente en el área de la hipermedia, debido a que los integrantes del equipo de desarrollo proceden de diferente áreas de conocimiento, incluyendo creadores, diseñadores, artistas e ingenieros, que a su vez deben comunicarse con los usuarios finales en un proceso participativo y por ello también es necesario contar con herramientas software que alivien este proceso cognitivo. Sin embargo la realidad es que el dominio hipermedia no cuenta aún con este tipo de herramientas, y de hecho, tal y como se discute en la siguiente sección, existen una serie de problemas específicos que dificultan su inmediata elaboración.

3.2. Problemática del uso de pdh

Para hacer accesible el conocimiento implícito en los patrones hipermedia es necesario contar con mecanismos que hagan explícita su existencia y justificación, así como su organización y sus interrelaciones, ayudando a los diseñadores inexpertos a reducir la curva de aprendizaje para resolver sus problemas como verdaderos expertos. Aunque se han realizado algunos intentos en este sentido, todavía se pueden encontrar algunas carencias, las cuales pueden dar la clave de la falta de aceptación de los patrones de diseño por parte de la comunidad de diseñadores hipermedia. Estas carencias afectan a:

C1. **Los criterios de categorización**. El problema crucial en el uso de los patrones de diseño, es precisamente saber en qué contexto y sobre qué problema es indicado aplicar un determinado patrón. Una primera aproximación podría ser la definición de unos criterios que permitiesen categorizar los patrones por dicho objetivo, de tal manera que su búsqueda se viese aliviada. De las aproximaciones presentadas en la sección 2.3.2, la que mejor se adapta a estos requisitos es la descrita en [Garzotto *et al.*, 1999], aunque debería ser ampliada para recoger patrones relacionados con otros aspectos de diseño, como la seguridad o la personalización. El multiespacio presentado en [Nanard y Nanard, 1999], aunque cubre todas las posibles categorías de patrones hipermedia, debido a la gran cantidad de ellas que aglutina más que clasificarlos, los proyecta para analizar las diferentes facetas de los patrones.

C2. **Los catálogos de patrones**. Unos criterios de categorización no son suficientes si no van a acompañados de un catálogo de patrones. Aunque existe una gran cantidad de patrones, entre hipermedia y web, y algunos presentados como colección de patrones [33], solamente en [Lyardet *et al.*, 1999] son presentados bajo una clasificación centrada en aspectos de diseño. Esta clasificación no cubre todos los aspectos de diseño de una aplicación hipermedia presentados en la sección 2.2.1 “Características de las aplicaciones hipermedia”, como son los aspectos de personalización y seguridad.

C3. **Los lenguajes de patrones**. Mientras que un catálogo de patrones proporciona soluciones a una colección de problemas y puede ser asequible para un diseñador experto, un lenguaje de patrones ofrece, además, una estructura de conocimiento que refleja las interrelaciones naturales entre los patrones, aportando una solución detalla a un problema de diseño de gran escala, en este caso el diseño de aplicaciones hipermedia,

sirviendo de guía a los diseñadores. En el dominio hipermedia no ha habido ninguna tentativa de crear un lenguaje de patrones, debido seguramente a las carencias de los puntos anteriores, ya que para poder acometer dicha empresa el conocimiento de los patrones debe ser analizado y organizado. Sin embargo, si se puede encontrar lenguajes de patrones específicos para aplicaciones web [Graham, 2003a], y más en concreto para aplicaciones de comercio electrónico en [van Duyne *et al.*, 2002], los cuales podrían servir de referencia.

C4. **Marco de integración en el proceso de desarrollo de los métodos de diseño**. El contexto que se propone en este trabajo para la aplicación de los patrones de diseño es mediante la combinación con los métodos de diseño, convirtiéndose en una guía que pueda asistir al diseñador o desarrollador durante el proceso de diseño de aplicaciones hipermedia. Aunque los patrones de diseño hipermedia son siempre presentados como una técnica complementaria a los métodos de diseño [Rossi *et al.*, 1997], todavía no existe un marco formal donde estén incorporados al proceso de diseño y que determine cuándo y cómo pueden ser usados.

C5. **No existe un formato único de descripción**. 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 (p.e. el problema y el contexto del patrón), lo que dificulta la comprensión e intencionalidad del mismo para usuarios noveles, hasta formatos más completos como el descrito en la sección 2.3.1 “Características de los patrones de diseño hipermedia”, en el que no se dan detalles de implementación. Todos los patrones deberían estar descritos utilizando la misma estructura o por lo menos compartir una estructura subyacente mínima para poder tener un catálogo de patrones uniforme cuya estructura pueda formalizarse.

C6. **No existe una homogeneidad en el nivel de descripción y en el vocabulario utilizados**. 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. En [German y Cowan, 2000] se pone de manifiesto la gran cantidad de patrones de diseño hipermedia publicados hasta la fecha que describen el mismo problema pero desde diferentes puntos de vista, porque como ya se ha mencionado, los aspectos de diseño de las aplicaciones hipermedia son altamente dependientes, lo que hace que diferentes autores, según su formación, describan el mismo problema. Además, hay

que unir la existencia de dos estilos narrativos. El primero engloba los patrones recopilados hasta [German y Cowan, 2000], los cuales se caracterizan por promover la reutilización de macro-estructuras de diseño dependientes de alguno de los modelos o métodos de diseño expuestos en la sección 2.2.2, lo que dificulta su uso tanto a los diseñadores expertos, que además de averiguar qué patrón aplicar deben traducirlo al vocabulario que utilice el método de diseño, como a los diseñadores noveles que más que dificultad lo que les puede provocar es rechazo. Este tipo de patrones, a cambio, pueden aplicarse de manera automática en el proceso de diseño. El segundo estilo narrativo consiste en un lenguaje más coloquial cercano al usuario de la aplicación final, y están orientados a la descripción de problemas de alto nivel de descripción, que aunque no pueden ser aplicados en los métodos de diseño, podrían ser automatizados para su exploración. Todas estas cuestiones conllevan problemas de ambigüedad semántica lo que dificulta el desarrollo de herramientas de soporte común.

C7. **No existe una formalización de patrones diseño hipermedia**. Hasta la fecha no ha habido ningún intento por formalizar los patrones hipermedia que permita el desarrollo de herramientas de soporte a la gestión y al uso de los patrones, seguramente debido a los problemas mencionados anteriormente sobre la falta de homogeneidad en este área. Aunque en el área de la ingeniería del software existen una serie de trabajos (véase la sección 2.1.7) cuyo objetivo es formalizar los aspectos estáticos y dinámicos de la orientación a objetos (clases, objetos, etc.), lo que no permitiría una utilización directa de los conceptos en la hipermedia (nodos, enlaces, anclas, etc.).

C8. **No existen herramientas de soporte**. Aunque no existen formalizaciones para los patrones hipermedia, sí se pueden encontrar algunos repositorios web y algunos ejemplos de aplicación automática en herramientas de diseño, como se mencionó en la sección 2.3.3. En estos repositorios se pueden encontrar los patrones organizados por algún criterio (p.e. interfaz/composición, estructura/navegación y orientados a contenido en [12]) y de manera hipertextual pero no existe un formato de intercambio de datos que permita la comunicación con herramientas de soporte a modo de librerías. De los métodos hipermedia, sólo OO-H y WebML tienen herramientas software *ad-hoc* que soportan patrones relacionados con aspectos de la navegación y presentación, pero no proporciona una arquitectura o modelo de aplicación para ser reutilizado en otras herramientas.

3.3. Requisitos para una integración de patrones en el proceso de diseño hipermedia

Los problemas a los que se tiene que enfrentar un diseñador de aplicaciones hipermedia para poder incorporar el conocimiento y la experiencia registrada en modo de patrones en el proceso de diseño de las aplicaciones hipermedia, como ha sido descrito en la sección 3.1, pone de manifiesto la necesidad de contar con una serie de herramientas que asistan al diseñador en ese proceso cognitivo de búsqueda, selección y aplicación tanto de forma manual como automatizada. Estas necesidades fueron plasmadas en la definición de los objetivos de esta tesis, en la sección 1.3. Retomando dichos objetivos y teniendo en cuenta los problemas y carencias específicos del dominio hipermedia, descritos en la sección 3.2, a continuación se incluyen una serie de requisitos que se deberían contemplar a la hora de materializar los objetivos:

O1. **Definir un marco cognitivo que organice el conocimiento subyacente a los patrones de diseño hipermedia mediante un conjunto de herramientas que asistan al diseñador**. Este marco cognitivo debe contar con:

   R1. Un conjunto de criterios que permita organizar los patrones existentes y futuros de acuerdo a la naturaleza del problema que resuelven, el modelado de las aplicaciones hipermedia, y los diferentes niveles de abstracción encontrados en la literatura de los patrones.

   R2. Un conjunto de patrones que cubra las categorías resultantes de los criterios definidos y que permita resolver problemas específicos de diseño hipermedia.

   R3. Una estructura de patrones que tenga como objetivo aportar una solución detallada a un problema de diseño de gran escala, como es el diseño de una aplicación hipermedia, considerando cada uno de sus aspectos de diseño.

   R4. Un mecanismo que defina el modo y los pasos a seguir para la integración de los patrones hipermedia dentro del proceso de diseño guiado por métodos, teniendo en cuenta la naturaleza conceptual de los patrones de diseño hipermedia.

Este conjunto de herramientas asistirá al usuario en la búsqueda, selección y aplicación de patrones, así como en la integración de los mismos en el proceso de diseño guiado por métodos. Sin embargo, para que este marco sea completo es necesario dotar a los patrones

de un formalismo que permita su representación computacional con el objeto de dar soporte a dichas herramientas de manera software.

O2. **Definir un marco conceptual que permita formalizar el conocimiento de los patrones de diseño hipermedia para ser tratados computacionalmente**. Este marco conceptual debe contar con:

R5. Un modelo de representación que permita formalizar todos los elementos que intervienen en la descripción de los patrones de diseño hipermedia de manera exhaustiva y completa. Este modelo debería contemplar ciertos requisitos:

R5.1. Debe representar el formato del patrón para mantener la descripción textual junto a la descripción formal y permitir así una automatización basada en la exploración.

R5.2. Debe representar las relaciones estructurales y asociativas existentes entre los patrones que permite la navegación a través de un conjunto de patrones relacionados.

R5.3. Debe representar los elementos propios de las aplicaciones hipermedia que permitan capturar el problema y la solución que plantea un patrón de diseño hipermedia con el objeto de automatizar su aplicación.

R5.4. Debe ser extensible para que pueda adaptarse a la heterogeneidad de formatos y vocabularios encontrados en los patrones de diseño hipermedia.

R5.5. Debe ser abstracto, independizándose del objeto de su aplicación software.

R5.6. Debe permitir el desarrollo de repositorios de patrones que sean comunes entre las posibles herramientas que se desarrollen para su gestión, exploración y aplicación.

R5.7. Debe ser posible la interoperabilidad entre los patrones representados con este modelo y su representación como entidades de diseño en los métodos hipermedia.

R5.8. Debe ser posible que los usuarios de los patrones de diseño hipermedia puedan formalizar sus patrones acorde a sus conocimientos.

Este modelo de representación sentará las bases para el tratamiento de los patrones por herramientas software tanto para su exploración como para su aplicación. Sin embargo con el objeto de validar su factibilidad técnica y verificar el cumplimiento de estos requisitos, es necesario el desarrollo de una serie de herramientas.

O3. **Definir un marco de implementación que demuestre que el marco conceptual definido permite el desarrollo de repositorios para la gestión, organización y recuperación de los patrones de manera compartida por herramientas software que automaticen las definidas en el marco cognitivo**. Para ello, este marco de implementación debe contar con:

R6. Una herramienta que permita la gestión de repositorios de patrones de diseño hipermedia en base al modelo de representación definido.

R7. Una herramienta que permita la exploración y búsquedas avanzadas de patrones en los repositorios desarrollados.

R8. Un módulo que permita integrar en una herramienta CASE de diseño de aplicaciones hipermedia, la aplicación de patrones para la resolución guiada de problemas de diseño.

Adicionalmente, y para que el marco cognitivo y conceptual sean aplicables en diferentes contextos, deben ser:

- Independientes de métodos y modelos específicos de diseño hipermedia.
- Flexibles a las posibles nuevas características que puedan tener los patrones futuros.

3.4. Resumen del capítulo

La elaboración de esta tesis surge de la necesidad de integrar el conocimiento y la experiencia registrados en modo de patrones en el proceso de diseño de las aplicaciones hipermedia debido a sus ya probados beneficios en otros dominios software, como se ha mostrado en el Capítulo 2 “Estado de la cuestión”. Sin embargo, tras el estudio realizado en la sección 3.1 sobre qué herramientas son necesarias para su aplicación tanto de manera individual, es decir contar con catálogos y lenguajes que asistan al diseñador en la búsqueda y selección de aquellos patrones que resuelvan un determinado problema de diseño, como en conjunción con los métodos, es decir contar con un mecanismo que permita adaptar el uso de patrones durante el proceso de diseño, se detectaron una serie de carencias y problemas.

A modo de resumen sobre las cuestiones tratadas en este capítulo y que van a perfilar los objetivos específicos de esta tesis, en la tabla 3.1 se muestra cómo dichos objetivos, y en concreto, sus requisitos van a cubrir las deficiencias enumeradas en la sección 3.2.

Objetivos	**Requisitos**	**Carencias**
O1. Definir un marco cognitivo	R1 R2 R3 R4	C1. Carencia de unos criterios de categorización C2. Carencia de un catálogo de patrones C3. Falta de un lenguaje de patrones C4. Falta de un marco de integración en el proceso de diseño de los métodos
O2. Definir un marco conceptual	R5.1, R5.2, R5.4 R5.3, R5.4 R5.5 R5.6, R5.7, R5.8	C5. No existe un formato único de descripción C6. No existe un vocabulario común C7. No existe una conceptualización de los patrones hipermedia
O3. Definir un marco de implementación	R6 R7 R8	C8. No existen herramientas de soporte

Tabla 3.1: Relación entre los objetivos y las deficiencias

Cada uno de estos objetivos específicos será desarrollado en los siguientes capítulos. El objetivo O1 será abordado en el Capítulo 4 “Integración de los patrones de diseño hipermedia en el proceso de diseño”, el objetivo O2 en el Capítulo 5 “Representación formal de los patrones de diseño hipermedia” y finalmente el O3 será abordado en el Capítulo 6 “Evaluación”.

Capítulo 4

Integración de los patrones de diseño hipermedia en el proceso de diseño

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, como se ha discutido en el capítulo anterior, esta reutilización no es tan inmediata como podría parecer, puesto que es preciso encontrar el patrón que se adapte a nuestro problema y adaptarlo a nuestro entorno. El objetivo de este capítulo es establecer un marco que permita aplicar dicho conocimiento a la resolución de problemas concretos del diseño hipermedia, convirtiendo a los patrones en guías activas del proceso de diseño. Es importante recordar que en el contexto de este trabajo se entiende el diseño como un proceso de ingeniería que es dirigido por algún método como los presentados en la sección 2.2.3 "Métodos de desarrollo", y no como un diseño artesanal dirigido por herramientas de autor como NetObjects' Fusion o DreamWeaver. Para que se produzca la sinergia entre patrones y proceso de diseño es necesario el desarrollo de una serie de mecanismos que organicen el conocimiento subyacente a los patrones de diseño hipermedia (pdh) hacia ese objetivo:

- Un conjunto de criterios a partir del cual los patrones puedan ser categorizados de acuerdo al contexto donde serán aplicados y la naturaleza del propio patrón (véase la sección 4.1).

- Una colección de soluciones a problemas de diseño hipermedia categorizados según los criterios propuestos (véase la sección 4.2).

- Un lenguaje de patrones, fruto de organizar las inter-conexiones existentes entre los diferentes patrones, con objeto de resolver un problema común, el diseño de una aplicación hipermedia (véase la sección 4.3).

- Un marco que integre los patrones en el proceso de diseño hipermedia para ayudar así al diseñador durante la resolución de problemas (véase la sección 4.4).

4.1. Criterios para la categorización de pdh

A la hora de categorizar los patrones se podría considerar que cada patrón es una categoría en sí misma, 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 patrones de diseño pueden ser clasificados atendiendo a diferentes criterios, como se ha mostrado tanto en el campo de la ingeniería del software (véase la sección 2.1.3) como en el campo de la hipermedia (véase la sección 2.3.2). Cada una de esas clasificaciones corresponde a una necesidad diferente del diseñador dictada por la situación concreta de diseño con la que se está tratando, dándose la circunstancia de la coexistencia de diferentes catálogos sobre el mismo conjunto de patrones pero destinados a diferentes propósitos. Por ejemplo, los patrones recogidos en [Gamma *et al.*, 1994] han sido clasificados hasta la fecha por su alcance y su propósito en [Gamma *et al.*, 1994] y según las relaciones existente entre los patrones en [Zimmer, 1995].

Los criterios que a continuación se presentan tienen como objetivo categorizar los patrones hipermedia de acuerdo a la naturaleza del problema que resuelven, **el modelado de las aplicaciones hipermedia**, y a los diferentes **niveles de abstracción** que se pueden encontrar en cómo se describen los pdh. Así mientras por un lado el diseñador podrá buscar patrones que resuelvan un determinado problema, también podrá localizar aquellos que mejor se adecuen al grado de detalle del diseño.

A la hora de seleccionar cuáles van a ser esos criterios, se han tenido en cuenta una serie de propiedades que deberían tener [Kardell, 1997]:

- **Ser universales**. Los criterios no deberían ser sólo extraídos a partir del estudio de los patrones existentes, sino también teniendo en cuenta cuál es el objetivo de dichos criterios, siendo así válidos para todos los patrones, tanto si ya han sido escritos como si no.
- **Ser conceptuales**. Un criterio debería permitir definir categorías que sean, conceptualmente, subconjuntos de ese criterio, proporcionando información más detallada al usuario sobre el problema general para el cual puede ser utilizado el patrón. De esta forma, el catálogo es más fácil de aprender y de usar.
- **Ser útiles**. Además de poseer las propiedades anteriormente mencionadas, el criterio debe reflejar un objetivo global de la categorización que sea relevante a sus usuarios y pueda determinar su utilidad.

Finalmente, en cuanto al conjunto de criterios y categorías que formarán la clasificación, es necesario **equilibrar su número** para que el conjunto de patrones por categoría sea lo suficientemente significativo como para evitar que la búsqueda de un patrón sea ineficiente y compleja.

Consecuentemente, considerando tanto las propiedades que deberían poseer los criterios como el objetivo global que debería tener el catálogo resultante, 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 del catálogo, que es resolver problemas durante el proceso de diseño de las aplicaciones hipermedia dentro del contexto de los métodos de diseño. A partir de las características expuestas en la sección 2.2.1, que describen las peculiaridades de las aplicaciones hipermedia, y del estudio de los métodos de diseño más relevantes presentados en la sección 2.2.3 que acometen el modelado de dichas características siguiendo un proceso sistemático, se han definido 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, que consiste en hacer explícitos los diferentes niveles de conceptualización encontrados durante el estudio de los patrones hipermedia en la sección 2.3.1, y que se corresponden con las siguientes categorías:

  - **Alto**. Patrones que proporcionan un conocimiento general de las características más relevantes de las aplicaciones hipermedia, utilizando para ello un vocabulario coloquial e independiente de los métodos de diseño, el cual podría se asequible para que los destinatarios de las aplicaciones puedan participar en el proceso de diseño de una manera más activa.
  - **Medio**. Patrones que proponen la reutilización de macro-estructuras que han sido reconocidas como buenas prácticas y que se encuentran cercanas al modelado conceptual. Utilizan un vocabulario dependiente de los métodos de diseño, por lo que están 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.

Estos criterios son semejantes a la combinación de los propuestos en [Garzotto *et al.*, 1999, Paolini y Garzotto, 1999] aunque con las siguientes diferencias. El primer criterio se corresponde al propuesto en [Garzotto *et al.*, 1999], pero sus categorías han sido ampliadas para recoger nuevos aspectos de diseño que han tomado una gran relevancia en los últimos años, como son la personalización y la seguridad. Con respecto al segundo criterio, tiene una semejanza a la idea propuesta en [Paolini y Garzotto, 1999], pero en vez de describir el mismo patrón en tres niveles de abstracción diferentes, se procede a categorizar los ya existentes según el propio nivel de abstracción que utilizó el autor del patrón para describirlo.

La figura 4.1 muestra el espacio de categorización producto de los dos criterios explicados anteriormente dispuestos como ejes ortogonales.

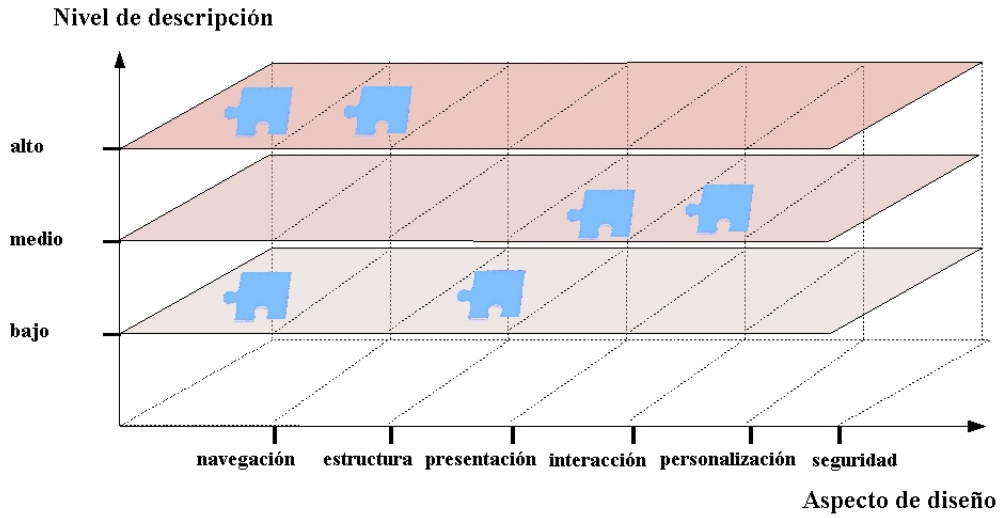


Figura 4.1: Un espacio de categorización para patrones de diseño hipermedia

El eje que representa el criterio **Nivel de descripción** proyecta tres planos paralelos sobre el eje **Aspecto de diseño**, es decir los patrones serán emplazados en alguno de estos planos y sólo en uno. Las categorías de **Aspecto de diseño** no son disjuntas debido a las interrelaciones existentes entre los diferentes aspectos de diseño involucrados en el diseño

de una aplicación hipermedia, como ya se mencionó en la sección 2.2.1. Aunque en la clasificación que se muestra en la sección 4.2 no hay ningún patrón que afecte a dos categoría, no se descarta que en el futuro pueda haberlo.

En la siguiente sección, se procederá a la categorización de los patrones hipermedia de acuerdo a esta clasificación.

4.2. Definición de un catálogo de pdh

Antes de proceder a mostrar los patrones que formarán parte del catálogo presentado en este trabajo, y siguiendo los criterios de organización propuestos en la sección anterior, se mencionan las restricciones que se han tenido en cuenta a la hora de seleccionar los patrones:

- **El patrón debe resolver problemas específicos de las aplicaciones hipermedia**. Existen ciertos patrones cuyo propósito es el de proporcionar aquellos mecanismos que dan soporte a la estructura y comportamiento hipermedia, como por ejemplo, el patrón `Node as a Navigational View` [Garrido *et al.*, 1997] que define un nodo hipermedia asociado a un objeto o grupo de objetos. Sin embargo, este tipo de patrones serían redundantes, puesto que partimos de la utilización de métodos de diseño para llevar a cabo el desarrollo de las aplicaciones hipermedia y son los propios métodos los que se encargan de aportar dichos mecanismos.

- **El patrón debe ser independiente del dominio de la aplicación**. Debemos puntualizar que no nos encontramos ante un dominio concreto de aplicación sino ante un paradigma como es el hipertexto, que es utilizado como tecnología subyacente en diferentes dominios como pueda ser el comercio o el aprendizaje electrónico, e incluso la propia Web puede ser vista como una implementación de este paradigma. Por lo tanto, hay patrones dependientes del dominio que no aportan una solución extrapolable a cualquier otra aplicación hipermedia. Por ejemplo, el patrón `Virtual Product` [German y Cowan, 1999] determina la información que debería aparecer en una tienda electrónica para un producto, un patrón que no es aplicable fuera del ámbito del comercio electrónico.

- **El patrón debe de resolver un problema de diseño conceptual**. El objetivo de este trabajo es integrar los patrones dentro del proceso de diseño de las aplicaciones hipermedia, teniendo como premisa fundamental que ese proceso debe estar dirigido por alguno de los métodos de diseño existentes. Los patrones que formarán parte de este catálogo deberán tratar con entidades de diseño que describan una solución lógica no basada en asuntos técnicos, sino en los requerimientos de una aplicación hipermedia. Por ejemplo, un patrón como `Node creation method` [Garrido *et al.*, 1997] 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, estaría fuera del alcance de este trabajo.

Por lo tanto a la hora de realizar este catálogo, de la extensa bibliografía consultada se han excluido aquellos patrones que recaen en alguna de las restricciones anteriormente mencionadas. Estas restricciones son necesarias para determinar el alcance de la clasificación descrita en la sección anterior y definir el tipo de patrones que los usuarios podrán encontrar en este catálogo, de manera que se incremente su potencial utilidad. Además, puesto que la hipermedia es un campo multidisciplinar también se han consultado patrones de diferentes dominios, como el de la seguridad o la interacción persona-ordenador, que podrían resultar de gran valor para resolver ciertos problemas para los cuales todavía no existen patrones propios.

A continuación se presentan los patrones seleccionados para formar parte del catálogo organizados en diferentes tablas para conseguir una mayor legibilidad. En el anexo A "Un lenguaje de patrones de diseño hipermedia" se incluye la descripción completa de los patrones con su correspondiente referencias bibliográficas.

Cada tabla se corresponde con cada una de las categorías del criterio **Nivel de descripción**, la tabla 4.1 enumera los patrones pertenecientes a la categoría de nivel de descripción **alto**, mientras que la tabla 4.2 recoge los de la categoría de nivel de descripción **medio** y la tabla 4.3 recoge los patrones de nivel de descripción **bajo**. En dichas tablas los patrones han sido clasificados por el **Aspecto de diseño** que la exposición del problema del patrón afronta. Es decir, el patrón es clasificado según la respuesta dada a la pregunta **¿en qué aspecto de diseño se puede enmarcar el problema que plantea el patrón?**, incluyendo junto a cada patrón una breve frase que resume su intención a modo de requerimiento funcional.

Finalmente, los nombres de los patrones se mantienen en inglés para conservar su principal cualidad de mecanismo de comunicación entre los diseñadores de la comunidad, además de preservar el sentido original que quería transmitir su autor. Además, por motivos de identificación se ha añadido 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, I-Interacción, Z-Personalización y S-Seguridad), y un número según su orden de aparición.

Navegación	
`[AN1] Multiple Ways to Navigate`	El usuario debe poder moverse por la información de diferentes modos, según su motivación o intención
Estructura	
`[AE1] User-centered Structure`	El usuario debe poder acceder a la información y a sus tareas de una manera intuitiva y sencilla
Presentación	
`[AP1]Aesthetic`	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] Interaction`	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] Personalization`	El usuario debe poder acceder a una aplicación adaptada a sus intereses y necesidades
Seguridad	
`[AS1]Access Control`	El usuario debe acceder de manera controlada al sistema y estar autorizado para realizar ciertas tareas

Tabla 4.1: Patrones de diseño hipermedia con un nivel de descripción alto

Navegación	
[MN1] Index Navigation	El usuario debe poder acceder a un determinado miembro de un grupo de nodos de manera directa
[MN2] Guided Tour Navigation	El usuario debe poder acceder a un grupo de nodos relacionados de manera sencilla
[MN3] Navigational Context	El usuario debe poder explorar de diferentes maneras un nodo según la tarea que esté realizando
Estructura	
[ME1]Hierarchical Organization	El usuario debe poder moverse con familiaridad por grandes cantidades de información
[ME2]Task-based Organization	El usuario debe poder completar un conjunto de tareas relacionadas de una manera rápida y sencilla
Presentación	
[MP1] Information-Interaction Decoupling	El usuario debe poder diferenciar entre contenidos y tipos de control (navegación y no-navegación)
[MP2] Information-Interaction Coupling	El usuario debe de reconocer qué controles están asociados con qué contenidos
[MP3]Behavioral Grouping	El usuario debe de reconocer los tipos de control que están relacionados
[MP4] Define and Run Presentation	El usuario debe percibir los elementos multimedia como una composición estética
[MP5] Synchronise Channels	El usuario debe percibir los elementos multimedia como una composición armónica
Interacción	
[MI1]Information on Demand	El usuario debe poder controlar la información extra que recibe en su pantalla
[MI2]Process Feed-back	El usuario debe ser informado del estado de la interacción de tal modo que sepa qué esperar
Personalización	
[MZ1]Role Subtype	Los usuarios deben estar organizados por tipos según sus similitudes

[MZ2] Structure Personalization	El usuario debe acceder sólo a aquellos nodos y contenidos que le interesan
[MZ3] Content Personalization	El usuario debe recibir los contenidos de manera personalizada
Seguridad	
[MS1] Authorization	El usuario debe de tener diferentes tipos de acceso a los recursos
[MS2] Role-Based Access Control	Cada tipo de usuario debe tener los derechos de acceso que le correspondan
[MS3] Multilevel Security	Cada usuario debe acceder sólo a aquellos recursos para los cuales tenga permisos

Tabla 4.2: Patrones de diseño hipermedia con un nivel de descripción medio

Navegación	
[BN1] Bread Crumbs	El usuario debe poder saber dónde se encuentra con respecto a la jerarquía de información
[BN2] Site Map	El usuario debe poder saber dónde se encuentra dentro de la estructura de información de la aplicación
[BN3]Search Action Module	El usuario debe poder encontrar un elemento o información específica de manera rápida
[BN4] One Jump Home	El usuario debe poder llegar al nodo de entrada fácilmente desde cualquier sitio
Estructura	
[BE1] Collection Center	El usuario debe saber qué tipo de información o tareas recoge un conjunto de nodos para poder seleccionar uno de ellos
[BE2] Node as a Single Unit	El usuario debe percibir que cada nodo es una unidad de información autocontenida y tiene sentido para él

[BE3] Homepage	El usuario debe ser consciente del objetivo de la aplicación y cómo puede moverse para realizar sus tareas
Presentación	
[BP1] Navigation Bar	El usuario debe encontrar siempre visibles y de manera consistente las herramientas de ayuda a la navegación
[BP2] Footer Bar	El usuario debe ser consciente de las condiciones de uso de la aplicación
Interacción	
[BI1] Action Button	El usuario debe ser consciente de la importación de la acción en relación a otras acciones que se puedan realizar
Personalización	
∄	
Seguridad	
∄	

Tabla 4.3: Patrones de diseño hipermedia con un nivel de descripción bajo

El conjunto de patrones aquí presentados forman lo que se denomina un catálogo. A modo de resumen, en la figura 4.2, se muestran todos los patrones del catálogo posicionados en su espacio de categorización. 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 concreto sobre la navegación de la aplicación, y desea saber qué patrones puede utilizar, puede seleccionar alguno de los que aparece en el eje **Aspecto de diseño**, e incluso según el nivel de detalle en el que quiera profundizar según su habilidad o conocimientos, puede reducir el número de patrones a la intersección con el eje **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.

Para finalizar, mencionar que este catálogo se ha implementado en una web como se describe en la sección 6.3.1.

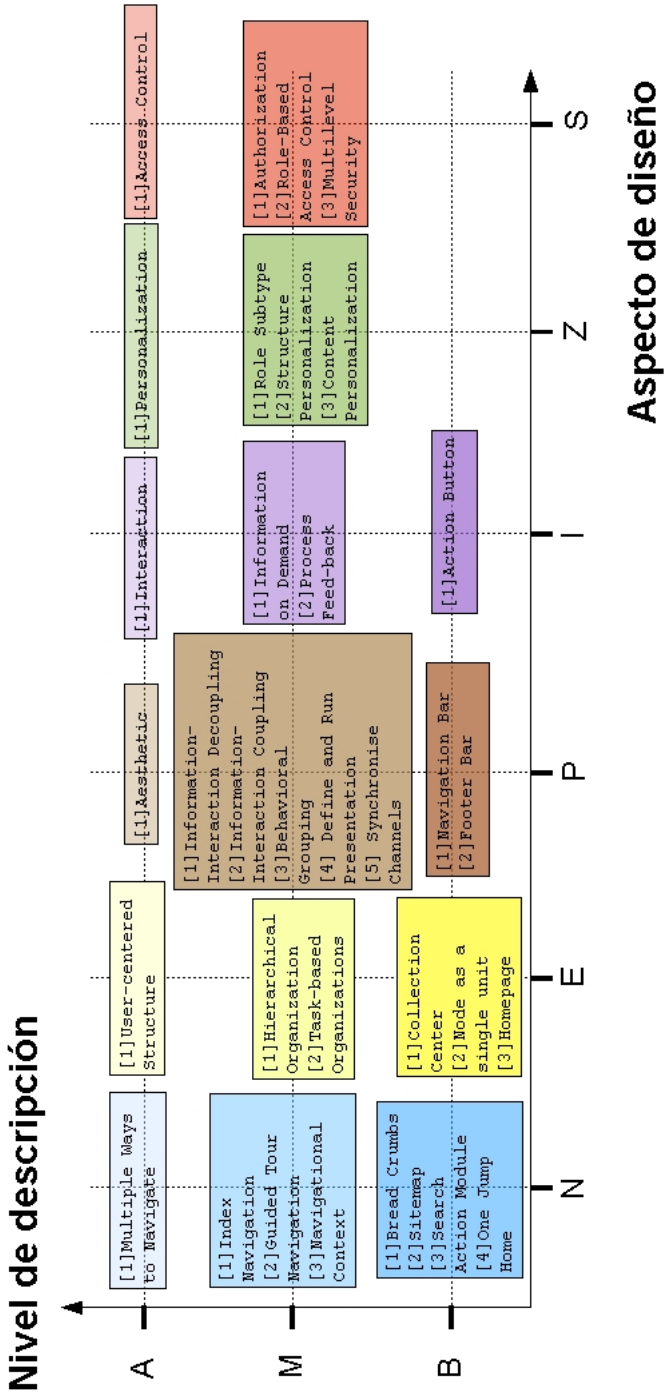


Figura 4.2: Un catálogo de patrones de diseño hipermedia

4.3. Definición de un lenguaje de pdh

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 del lenguaje utilizado para hacer llegar su conocimiento al usuario, con el único objeto 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 como [12, 33]. 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.

Al igual que Alexander en [Alexander *et al.*, 1977], puso énfasis en la palabra *"A"* de *"A pattern language"*, el lenguaje que se presenta en esta sección, es *un lenguaje*, y no pretende en ningún momento ser *el lenguaje*. El objetivo de este lenguaje de patrones, en concreto, 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, ayudando así al usuario a seleccionar el conjunto de patrones que le permitan desarrollar el núcleo de la aplicación hipermedia ajustándose a sus necesidades. Por lo tanto, este lenguaje podría colaborar con otros lenguajes que tratasen aspectos de diseño más especializados, como por ejemplo patrones para aplicaciones hipermedia ubicuas o formar parte de otros lenguajes de mayor ámbito pertenecientes a diferentes dominios de aplicación como puedan ser los patrones para aplicaciones hipermedia de comercio electrónico como puede verse en la figura 4.3. Además, el conocimiento de los patrones crece día a día, y al igual que nuestro propio lenguaje es algo dinámico que va evolucionando y adaptándose con el tiempo, el lenguaje de patrones de diseño hipermedia aquí presentado no está exento de los posibles cambios que, en un futuro, la incorporación de nuevos patrones pueda provocar en su estructura y organización.

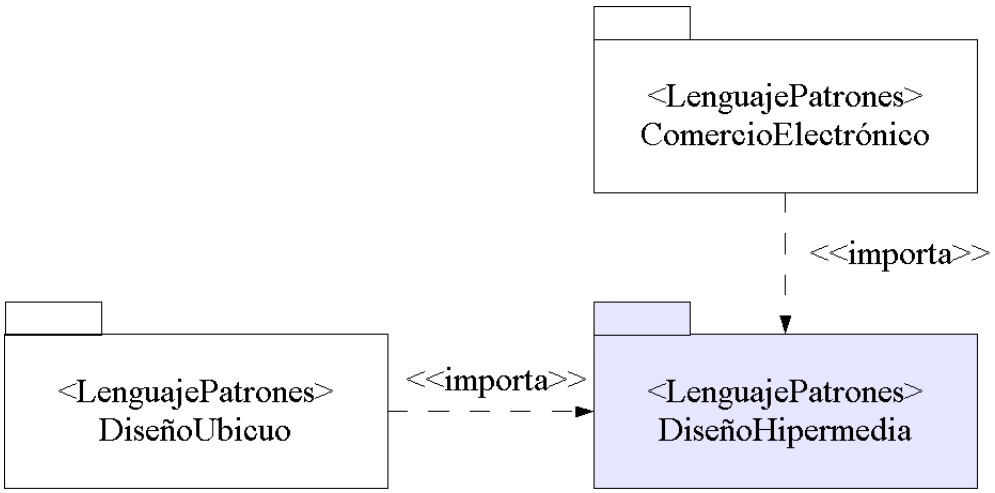


Figura 4.3: Colaborando con otros lenguajes de patrones

El lenguaje resultante es fruto del estudio de las posibles relaciones de colaboración que deberían existir entre los diferentes patrones incluidos en el catálogo de la sección 4.2 y cuyas posibles combinaciones diesen como resultado una solución de diseño coherente con el objetivo del lenguaje. Este trabajo previo fue necesario debido a que dichos patrones han sido tomados de diferentes fuentes bibliográficas y de diferente autores, por lo que en algunas ocasiones ha sido necesario adaptar el contenido del patrón, aunque no su esencia, y otras veces ha llevado a la reestructuración de los patrones seleccionados. Finalmente, los patrones han sido organizados en grupos conceptuales que comparten un objetivo común dentro del diseño de una aplicación hipermedia, en este caso cada grupo dirige la resolución de los diferentes problemas que se pueden encontrar al abordar cada uno de los aspectos de diseño presentados en la sección 2.2.1, haciendo así más explícito la motivación y el alcance de este lenguaje, que como ya se ha mencionado es la de servir de guía en el diseño de aplicaciones hipermedia basado en métodos.

Cada grupo es brevemente introducido por un patrón de mayor nivel (`[AE1]User-centered Structure`, `[AN1]Multiple Ways to Navigate`, `[AP1]Aesthetic`, `[AI1]Interaction`, `[AZ1]Personalization`, `[AS1]Access Control`), que ha sido elaborado expresamente a partir de diferentes patrones para recoger mejor la esencia del grupo, es decir, los diferente problemas a los que se pretende dar solución, y dar una mayor cohesión en la solución propuesta, es decir, cómo la combinación de los patrones de menor escala guían al diseñador a afrontar cada uno de los proble-

mas propuestos. Finalmente, el lenguaje parte de un patrón genérico, `[A]Hypermedia Application`, encargado de explicar esta división y la motivación de cada uno de esos grupos de patrones, teniendo como principal hilo conductor describir las principales características que hacen a las aplicaciones hipermedia diferentes de cualquier otra aplicación software y por qué deben ser tenidas en cuenta a la hora de abordar su desarrollo. Este patrón, al igual que los representantes de cada grupo, ha sido elaborado especialmente con el propósito de dar una visión general del alcance del presente lenguaje, introduciendo a qué tipos de problemas de diseño comunes a las aplicaciones hipermedia pretende dar solución.

A continuación se presenta el lenguaje de manera resumida actuando a la vez como sumario e índice, aunque puede ser consultado de manera detallada en el apéndice A.

`[A]Hypermedia Application`

Si la aplicación a 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.

¿Cómo afrontar el diseño de una aplicación hipermedia para que sea útil y fácil de usar por el usuario?

- El patrón `[AE1]User-centered Structure` ayuda a organizar la información teniendo en cuenta las necesidades de los usuarios.
- El patrón `[AN1]Multiple Ways to Navigate` guía al usuario tanto a la hora de explorar el espacio de información, como al acometer sus tareas o encontrar lo que busca de manera rápida.
- El patrón `[AP1]Aesthetic` 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.
- El patrón `[AI1]Interaction` mejora la comunicación entre el usuario y la aplicación.

- El patrón [AZ1]Personalization ayuda a que la aplicación sea utilizada por el mayor número de usuarios.
- El patrón [AS1]Access Control controla quién puede hacer qué cosas en la aplicación.

[AE1]User-centered Structure

Se desea reflejar la estructura del dominio de una aplicación hipermedia.

¿Cómo estructurar la información y las tareas de la aplicación para proporcionar un acceso intuitivo y sencillo al usuario?

- El patrón [ME1]Hierarchical Organization organiza la información de manera jerárquica para permitir la navegación por grandes cantidades de información.
- El patrón [ME2]Task-Based Organization organiza las tareas relacionadas para que puedan ser completadas.
- El patrón [BE1]Collection Center presenta cada subcategoría de la estructura o conjunto de tareas.
- El patrón [BE2]Node as a Single Unit define lo que sería cada una de las unidades de información de la estructura.
- El patrón [BE3]Home Page se puede utilizar como nivel más alto de la estructura organizativa de la aplicación.

[AN1] Multiple Ways to Navigate

Ahora, es necesario dotar al usuario de mecanismos de navegación y de acceso al espacio de información.

¿Cómo hacer que el usuario pueda moverse por la información de diferentes modos, según su motivación o intención de manera clara y consistente a la vez que se le facilita encontrar la información deseada?

- El patrón [MN1]Index Navigation ayuda a los usuarios a seleccionar un destino.

- El patrón `[MN2]Guided Tour Navigation` ayuda al usuario a completar sus tareas de manera guiada.
- El patrón `[MN3]Navigational Context` hace que el espacio de navegación pueda ser explorado por diferentes criterios o contextos.
- El patrón `[BN1]Location Bread Crumbs` orienta al usuario sobre el estado actual de su navegación actual con respecto a una rama del espacio de información permitiéndole volver a un punto anterior.
- El patrón `[BN2]Site Map` muestra la situación del usuario con respecto al espacio total de información de la aplicación y le permite acceder de manera directa a cualquier punto.
- El patrón `[BN3]Search Action Module` ayuda al usuario a conseguir la información deseada de forma directa mediante consultas.
- El patrón `[BN4]One Jump Home` ayuda al usuario a volver en cualquier momento a la página principal o a otros puntos mayores de navegación en un sólo paso.

`[AP1]Aesthetic`

La apariencia de cada una de las unidades de información ya definidas debe ser especificada.

¿Cómo organizar la interfaz de la aplicación para que sea consistente e intuitiva a la vez que los contenidos son mostrados de manera estética?

- El patrón `[BP1]Navigation Bar` organiza las diferentes herramientas de accesos a la información de manera que estén siempre visibles y consistentes a través de todo el espacio de información.
- El patrón `[MP1]Information-Interaction Decoupling` ayudar al usuario a diferenciar entre contenidos y tipos de control (navegación y no- navegación).
- El patrón `[MP2]Information-Interaction Coupling` organiza los controles junto con los contenidos a los que está asociado.

- El patrón `[MP3]Behavioral Grouping` organiza juntos aquellos controles que están relacionados.
- El patrón `[MP4]Define and Run Presentation` muestra los elementos multimedia como una composición estética.
- El patrón `[MP5]Synchronise Channels` muestra los elementos multimedia como una composición armónica.
- El patrón `[BP2]Footer Bar` organiza el acceso a aquellos elementos secundarios como las condiciones de uso legales.

`[AI1]Interaction`

Uno de los modos de interacción es la navegación, pero es necesario mejorar la comunicación entre el usuario y la aplicación.

¿Cómo hacer que el usuario sienta que conoce y controla el proceso de interacción con respecto a la información y a las tareas que proporciona la aplicación?

- El patrón `[MI1]Information on Demand` hace que el usuario sea el que controle cuánta información desea visualizar en un nodo.
- El patrón `[MI2]Process Feed-back` informa al usuario del estado de la interacción de tal modo que sepa qué esperar en cada momento.
- El patrón `[BI1]Action Button` presenta aquellos elementos cuya activación provocan la ejecución de una acción.

`[AZ1]Personalization`

Hasta ahora se ha definido una estructura centrada en el usuario, diferentes tipos de acceso, el aspecto estético y cognitivo de los contenidos y cómo responder a la interacción con el usuario, todo ello separando cada uno de los aspectos de diseño. Ahora se desea considerar adaptar la aplicación a las necesidades de cada usuario.

¿Cómo hacer para que el usuario pueda acceder a una aplicación adaptada a sus intereses y necesidades?

- El patrón `[MZ1]Role Subtype` define los diferentes tipos de usuarios de la aplicación.
- El patrón `[MZ2]Structure Personalization` restringe el espacio de visualización a los intereses de cada tipo de usuario.
- El patrón `[MZ3]Content Personalization` personaliza los valores de los contenidos para cada tipo de usuario.

`[AS1]Access Control`

Finalmente si se está diseñando una aplicación hipermedia donde existen diferentes tipos de usuarios, tanto la información como los servicios definidos deberían tener un acceso controlado.

¿Cómo hacer para que el usuario pueda acceder de manera controlada al sistema y esté autorizado para realizar ciertas tareas?

- El patrón `[MS1]Authorization` define el control de acceso a los recursos.
- El patrón `[MS2]Role-Based Access Control (RBAC)` especializa el patrón anterior utilizando roles de usuario.
- El patrón `[MS3]Multilevel Security` permite asignar diferentes niveles de manipulación a la información.

Al igual que las palabras sin unas reglas de conexión no pueden formar un lenguaje, son las reglas de conectividad entre los patrones que conforman el lenguaje las que aportan el valor añadido con respecto a un catálogo, ayudando al diseñador a moverse de problema en problema de una manera efectiva. Estas relaciones se han extraído a partir del estudio de los componentes internos de cada patrón. En la figura 4.4 se muestra el lenguaje de patrones hipermedia a modo de mapa.

Cada patrón aparece conectado mediante flechas con aquellos patrones con los cuales guarda una relación estructural o de asociación, indicando cómo los patrones se combinan. En el margen inferior izquierdo de la figura 4.4 puede verse la leyenda de los diferentes tipos de línea utilizados. Los tipos de relación contemplados son:

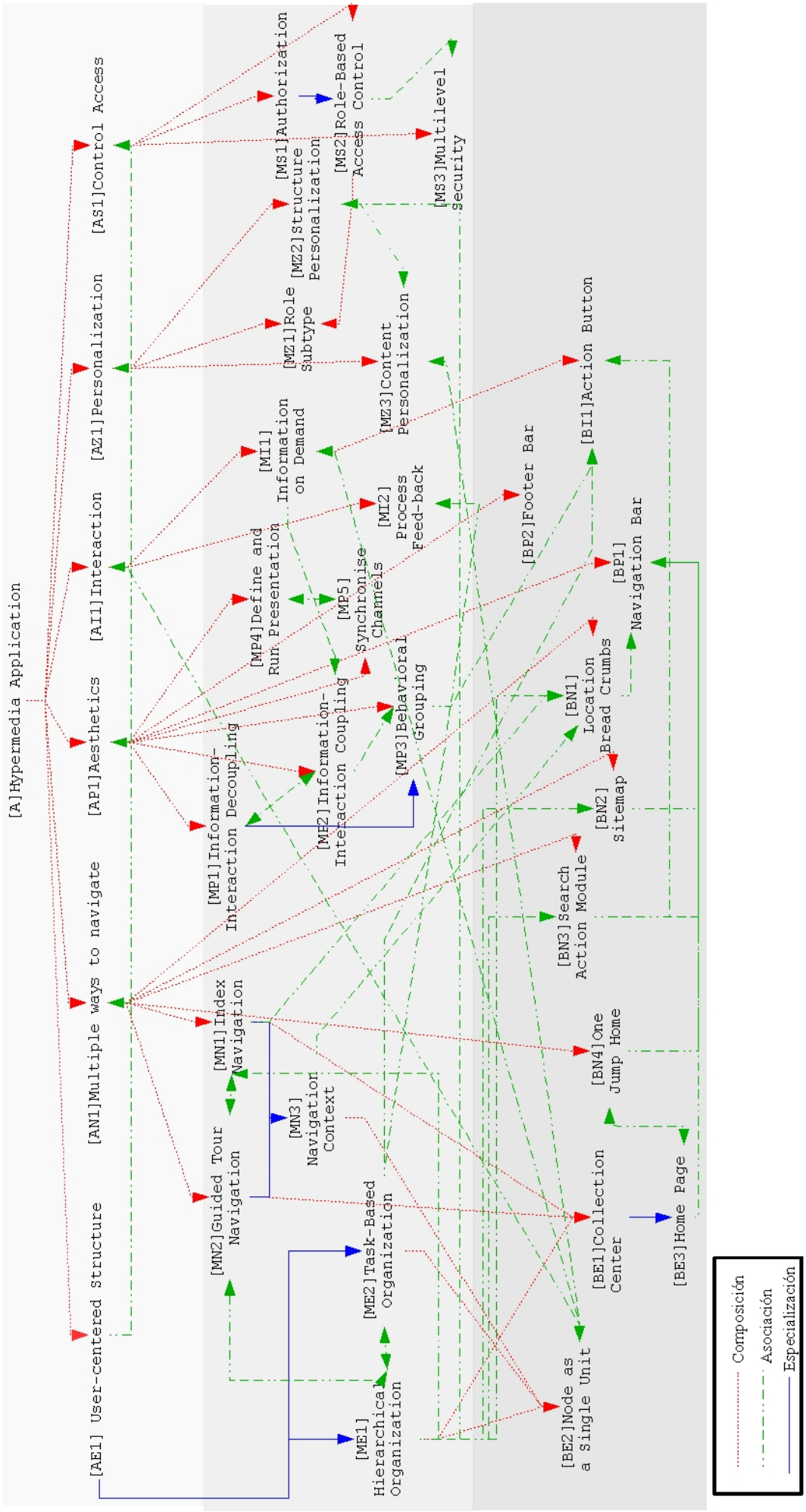


Figura 4.4: Mapa del lenguaje de patrones

- ..... X utiliza a Y. La solución de Y representa una parte de la solución de X (similar a la relación de composición de la orientación a objetos). Por ejemplo, el patrón `[MI1]Information on Demand` (X) utiliza la solución del patrón `[BI1]Action Button` (Y) para indicar al usuario cuál es el control que tiene que emplear para decidir la cantidad de información a visualizar en un nodo.
- . . . . . X asociado a Y. X sugiere un nuevo problema que es tratado por Y (similar a la relación de asociación de la orientación a objetos). Por ejemplo, el patrón `[BE1]Node as a Single Unit` (X) sugiere que si el nodo contiene demasiados elementos de información se puede utilizar el patrón `[MI1]Information on Demand` (Y) para que sea el usuario quien decida cuanta información desea visualizar.
- _____ X refina a Y. X trata con una especialización del problema que aborda Y. Por ejemplo, el patrón `[MS2]Role-Based Access Control` (X) es una especialización de `[MS1]Authorization` (Y) en la que el control de acceso estaría basado en roles.

Como ya se ha mencionado en sucesivas ocasiones, cada patrón es un plan para resolver un determinado problema, sin embargo la combinación de estos patrones contiene información organizativa que hace que el todo sea mayor que la suma de las partes. Cada una de las conexiones que salen del patrón raíz `[A]Hypermedia Application` trata un problema específico de diseño de las aplicaciones hipermedia (estructura, navegación, presentación, interacción, personalización y seguridad). Por lo tanto, a la hora de trabajar con el lenguaje de patrones, el usuario realiza un proceso iterativo, seleccionando alguno de los caminos propuestos y aquellos patrones de menor escala que resuelvan el problema planteado (conexiones de especialización y composición), pero además puede incluir otros patrones que planteen nuevos problemas (conexiones de asociación), hasta llegar a describir un diseño concreto en términos de los elementos que la constituyen y su configuración óptima. Para generar este diseño concreto se puede utilizar el procedimiento descrito por Alexander y que fue recogido en la sección 4.3 “Los lenguajes de patrones”. Un ejemplo de los posibles caminos existentes en este lenguaje queda recogido en la figura 4.5. Partiendo de la raíz de este lenguaje, el patrón `[A]Hypermedia Application`, propone que uno de los aspectos de diseño a tener en cuenta a la hora de enfrentarse al desarrollo de este tipo de aplicaciones debería ser organizar el espacio de información teniendo en cuenta la perspectiva del usuario, ayudando así a que el acceso a la misma se realice de manera más intuitiva y sencilla, sugiriendo la aplicación del patrón `[AE1] User-centered Structure`. Este

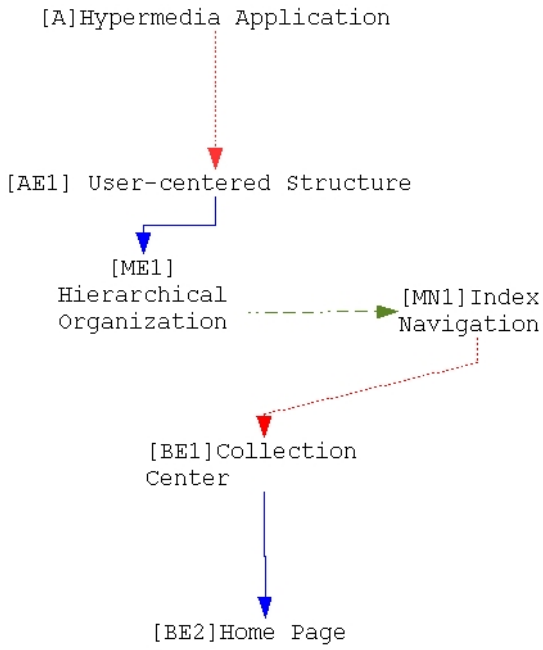


Figura 4.5: Ejemplo de un camino generado por el lenguaje

patrón se especializa a su vez en el patrón `[ME1]Hierarchical Organization` para organizar dicha información a modo de jerarquía, sugiriendo que se combine con una estructura de navegación basada en índices `[MN1]Index Navigation`. Este último patrón necesita del patrón `[BE1]Collection Center` para poder definir el punto de acceso que se utilizará como índice. Finalmente, la última relación que aparece en la figura 4.5 sugiere la existencia de un patrón que especializa al patrón `[BE1]Collection Center`, permitiendo aplicar el patrón `[MN1]Index Navigation` a nivel global de la aplicación, es decir que el nodo principal de entrada a la aplicación a a la vez actúe de índice.

4.4. Integración de los pdh en el proceso de diseño

Los métodos de diseño hipermedia descritos en la sección 2.2.3 tienen la cualidad de guiar al diseñador en su tarea de modelar diferentes aspectos de las aplicaciones hipermedia como la navegación, el dominio de la aplicación, la presentación, el comportamiento e incluso la seguridad a través de un conjunto de actividades y fases que permiten tratar 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 ya resueltos en el pasado, dando como resultado un esfuerzo duplicado, e incluso un diseño inconsistente y sistemas frágiles y de pobre trazabilidad si el diseñador no cuenta con la experiencia necesaria. Un enfoque más efectivo es construir un sistema hipermedia a partir de patrones bien documentados. Los patrones no son ni un método de desarrollo software ni un modelo de proceso pero ayudan a aliviar la complejidad del software en las diferentes fases de su ciclo de vida, complementando tanto a los métodos como a los procesos existentes. El catálogo de patrones presentado en la sección 4.2 documenta las mejores prácticas de diseño hipermedia, capturando las propiedades esenciales comunes a los mejores ejemplos de diseño, los cuales están basados en la experiencia de los diseñadores y desarrolladores para alcanzar aplicaciones útiles y usables. 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 4.3 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.

Los patrones de diseño hipermedia siempre se han propuesto como una técnica complementaria a los métodos, bien para solventar el hueco existente entre el modelado conceptual y la implementación del sistema, debido a que los patrones permiten hacer una correspondencia entre el problema y la solución [Garzotto *et al.*, 1999]; o bien, para enriquecer dichos métodos con primitivas de más alto nivel de abstracción, puesto que los patrones permiten expresar conocimiento con una única notación [Lyardet *et al.*, 1999]. Esta última propuesta no será considerada como base para la propuesta que aquí se expone, ya que el dotar a los métodos con un conjunto de primitivas que les permita definir un patrón como parte de su lenguaje de modelado, como es el caso de UML [Dong, 2003], tendría como resultado un trabajo individual para cada uno de ellos [1] debido a que no comparten ni los elementos gráficos que conforman sus productos ni los modelos de referencia en los cuales se basan dichos elementos, como se ha visto en la sección 2.2.3.

Teniendo en cuenta estas dos cuestiones, 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. En el proceso de desarrollo de una aplicación hipermedia, como aparece en la fi-

[1] En [Rossi y Lyardet, 1999] puede verse un ejemplo de como el patrón `Navigational Context` se encuentra incluido como primitiva de diseño dentro del método OOHDM

gura 4.6 basada en la figura 2.3, durante la fase de análisis de la aplicación se identifican las necesidades del usuario que conforman el espacio del problema y que son documentadas como requisitos. Posteriormente, en la fase de diseño 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 la dificultad del paso entre estos dos estados. Puesto que los patrones son presentados, como una regla que guía la transformación entre un problema dado y una solución, se puede establecer una correspondencia entre problemas conceptuales y soluciones de modelado. Este razonamiento puede verse reflejado en la figura 4.6 en el hecho de que los requisitos de la aplicación sean identificados con la parte del patrón (P), correspondiente al problema abordado, mientras que la solución (S) genera un modelo de diseño concreto.

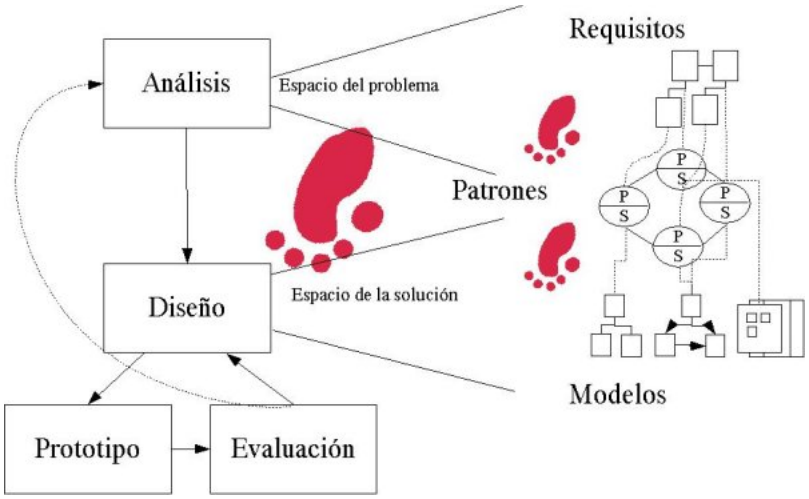


Figura 4.6: Integración de los patrones en el proceso de desarrollo de las aplicaciones hipermedia

En la figura 4.7 se muestra el diagrama de flujo que sigue el proceso de integración para que los requisitos guíen una aproximación basada en patrones y afrontar así el modelado conceptual:

Paso 1. Obtención de los requisitos software usando, por ejemplo, alguna de las técnicas propuestas en [Escalona y Koch, 2003]. Este primer paso es de preparación al proceso en sí y tiene como resultado la generación del documento con la especificación de requisitos de la aplicación a desarrollar.

Paso 2. Asignar a cada requisito funcional o de usabilidad 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 sección 4.2, las cuales incluyen un resumen de la intención de cada patrón. Los patrones seleccionados deberían recaer en los niveles de descripción Medio y Bajo, ya que son dichos patrones los que capturan soluciones aplicables a modelos conceptuales. Como resultado de este paso se obtiene una relación entre el identificador de requisito e identificador de patrón que mejor lo cubre.

Paso 2.1. Si alguno de los patrones seleccionados pertenece a 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 cuyo 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 sección 4.3.

Paso 3.1. A partir del campo “Relacionado con” del patrón, seleccionar todos aquellos patrones de menor escala que soportan a ese patrón, es decir seguir las relaciones de composición que se muestran en el mapa del lenguaje de la figura 4.4 e incluirlos junto al seleccionado en el Paso 2, ya que son necesarios para completar su solución.

Paso 3.2. Con respecto a las relaciones de especialización, sopesar si deben reemplazar al patrón seleccionado. En caso afirmativo, es necesario volver a repetir el Paso 3.

Paso 3.3. Finalmente, estudiar si los patrones que se encuentran relacionados por asociación en el campo “Relacionado con” y los que aparecen en el campo “Contexto”, pueden se de utilidad para completar al requisito . En caso afirmativo será necesario volver al Paso 1 por que puede llevar asociado la redefinición del mismo o la inclusión de uno nuevo requisito.

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. Como resultado de este paso se obtiene una relación entre identificador de requisito y los identificadores de los patrones que lo cubren.

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. Como resultado de este paso se obtiene la relación de identificador de requisito e identificadores de patrón organizados por aspecto de diseño.

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, dando como resultado un esquema de solución independiente de métodos de diseño.

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, dando como resultado un esquema de solución dependiente del método de diseño con el cual se vaya a generar una primera aproximación del 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. Además, es necesario mencionar que este proceso no pretende ser ortogonal al uso común de un lenguaje de patrones, como se mostró en la sección 4.3, sino que pueda ser utilizado de manera complementaria cuando el diseñador se enfrente a un nuevo desarrollo de una aplicación hipermedia a partir de un conjunto de requisitos identificados guiado por algún método de diseño.

Esta asociación de **patrones - requisitos** puede proporcionar las siguientes cualidades deseables en una especificación final de requisitos [Lowe y Hall, 1999]:

- **Verificable**: La aplicación de un patrón deja constancia documentada de que su requisito asociado ha sido tenido en cuenta.
- **Modificable**: Esa misma constancia de la aplicación de un determinado patrón permite realizar cambios en los requisitos de manera integral entre las fases de análisis y diseño, al ser identificado con entidades concretas de diseño.
- **Trazable**: La asociación del requisito con su estructura conceptual a través de un patrón permite tener constancia de cómo se ha resuelto el problema.

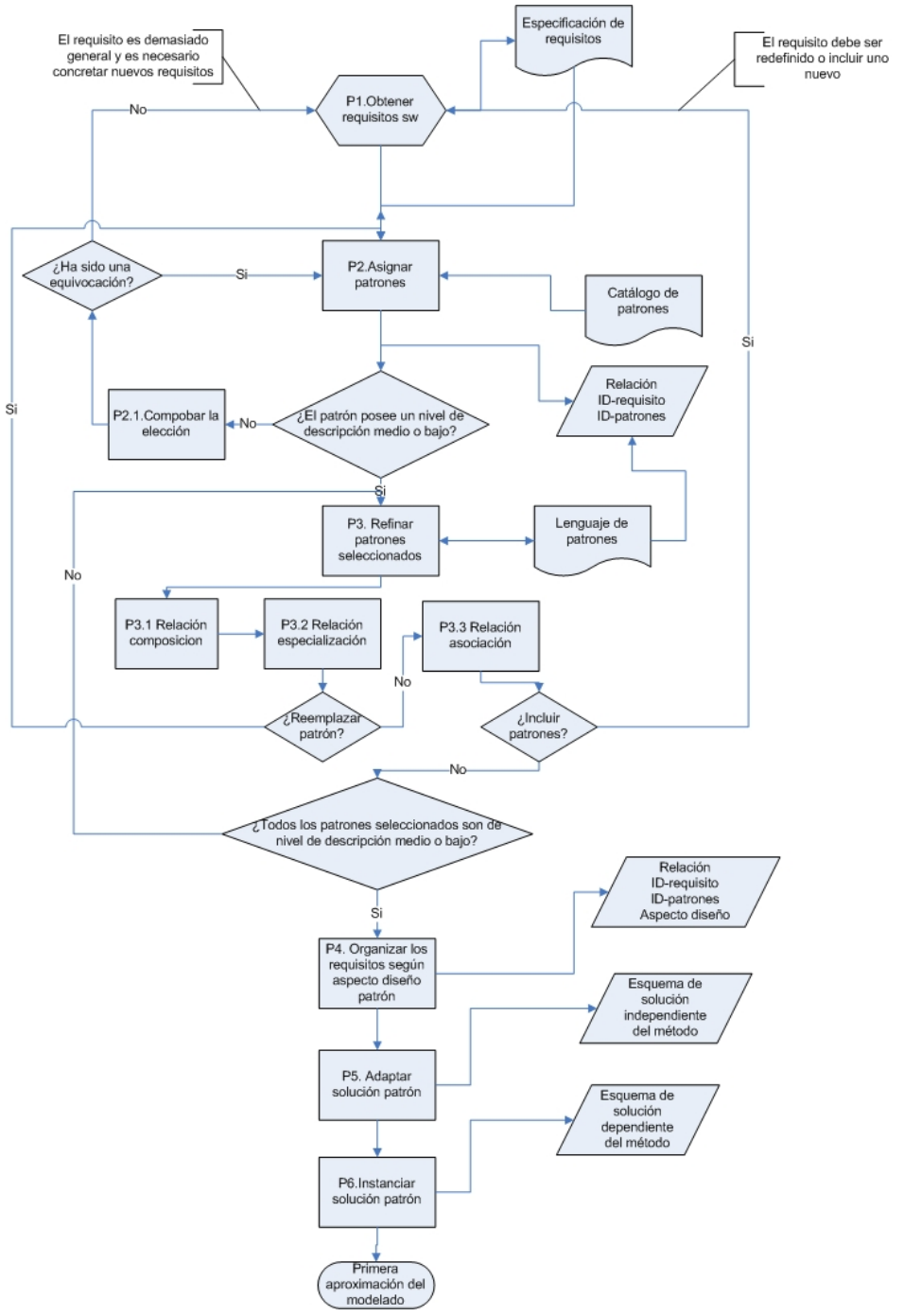


Figura 4.7: Diagrama de flujo del proceso de integración de los patrones en el proceso de diseño de los métodos hipermedia

4.5. Resumen del capítulo

En este capítulo se han identificado y se han elaborado una serie de herramientas como solución a los principales obstáculos que dificultan el acceso y la utilización de los patrones de diseño dentro del proceso de desarrollo de las aplicaciones hipermedia, y que pueden ser causa de su poca aceptación por parte de los diseñadores hipermedia, a pesar de sus probados beneficios. Estas herramientas son:

- **Unos criterios de clasificación** que organizan los patrones acorde a los diferentes aspectos de diseño a los que se enfrenta un diseñador hipermedia, así como a los diferentes niveles de abstracción encontrados en la literatura de los patrones, ayudando a organizar el espacio de patrones existente y el futuro.
- **Un catálogo de pdh** destinado a facilitar la búsqueda de patrones en la resolución de problemas concretos de diseño, organizado según los criterios propuestos.
- **Un lenguaje de patrones** orientado a ser una guía 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.
- **Un proceso de integración** que permite la aplicación de los patrones durante el proceso de diseño guiado por métodos ayudando a generar una primera aproximación al modelado conceptual a través de las soluciones propuestas en los patrones.

Dichos mecanismos permitirán asistir y guiar a los diseñadores y desarrolladores de aplicaciones hipermedia en el uso de los patrones tanto de forma individual como de manera integrada con los métodos 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 diferentes programas software de una manera integrada. Estas y otras cuestiones sobre la formalización de los patrones serán tratadas en el siguiente capítulo.

Capítulo 5

Representación formal de los patrones de diseño hipermedia

Aparte del catálogo y del lenguaje de patrones presentados en el capítulo anterior, la realidad es que los patrones hipermedia existentes se encuentran repartidos entre diferente publicaciones. Algunos de ellos se pueden encontrar accesibles a través de sitios web, pero tan sólo incluyen como principal funcionalidad navegar por su descripción textual, sin ninguna otra funcionalidad que les pueda aportar un valor añadido a este formato, como pueda ser permitir búsquedas avanzadas sobre diferentes campos del patrón que podrían dar como resultado la selección de uno de ellos tanto por el problema que platea como por el nivel de descripción con que lo hace. Estas representaciones pasivas hacen que el conocimiento que puedan tener los diseñadores sobre la existencia de estos recursos y su entendimiento sobre cuándo deberían ser aplicados sean los dos únicos condicionantes que determinan el éxito o el fracaso del uso de los patrones. Dada la gran cantidad de patrones de diseño hipermedia existentes, unos 200, los cuales pueden ser aplicados en diferentes contextos, y teniendo en cuenta la heterogeneidad de conocimientos y habilidades del equipo de desarrollo hipermedia, esta solución puede no ser del todo deseable a la hora de afrontar **qué patrón o patrones pueden aplicarse a un determinado problema**, **cómo adaptar un patrón al contexto donde quiere ser aplicado** y **cuál es el siguiente patrón que podría ser aplicado**, puesto que es el usuario el que debe:

- recorrer cada uno de los recursos, tanto web como bibliográficos, de manera individual;
- intentar recuperar aquellos patrones relevantes basándose bien en los descriptores, si los hubiese, que han utilizado los autores de los patrones para su organización, o en el nombre del patrón;
- por cada patrón recuperado, leerlo detenidamente y decidir si es adecuado;
- interpretar y adaptar la solución planteada a su entorno de diseño;
- y, finalmente, comprobar si con ello se ha satisfecho completamente la necesidad que originó la utilización del patrón.

La representación textual de los patrones compromete su accesibilidad y usabilidad, sobre todo para aquellos usuarios más noveles. En primer lugar, es necesario el desarrollo de repositorios que permitan la gestión, organización y recuperación de los patrones, así como la conexión entre diferentes repositorios para una recuperación más efectiva. En segundo lugar, es necesario contar con una interfaz de acceso a esos repositorios, siendo la Web Semántica [Berners-Lee *et al.*, 2001] el escenario idóneo puesto que los datos son procesados a partir de su significado, permitiendo al usuario realizar búsquedas más concretas sobre el contenido de los campos del patrón e incluso una exploración basada en la resolución de un problema concreto de diseño. Finalmente, sería positivo contar con herramientas que ayuden a los usuarios a realizar las actividades cognitivas involucradas en la aplicación de los patrones e incluso que faciliten su aplicación automática, siempre que sean susceptibles de ser automatizados, en combinación con herramientas de soporte al proceso de diseño de las aplicaciones. Todo ello podría influir en una mejor difusión, aceptación, desarrollo y empleo de los patrones.

El objetivo de este capítulo es presentar un modelo de representación formal para los patrones de diseño hipermedia (pdh) a partir del cual se puedan desarrollar herramientas como las mencionadas en el párrafo anterior. Este formalismo se basa en la representación del conocimiento capturado por los patrones de diseño hipermedia mediante el uso de ontologías (véase la sección 5.1). La arquitectura de dicho modelo (véase la sección 5.2) está formada tanto por la descripción estructural del patrón (véase la sección 5.2.1) como por la conceptualización del dominio para el cual fueron escritos, es decir, el dominio hipermedia (véase la sección 5.2.2). La fusión de ambas formalizaciones dará como resultado la

representación conceptual de un patrón de diseño hipermedia (véase la sección 5.2.3). Aunque esta formalización no puede reemplazar el entendimiento que aporta la descripción del lenguaje natural, si se puede pensar en ella como una representación complementaria que puede ser compartida y reutilizada entre personas y sistemas computacionales. Para ello, se ha utilizado las anotaciones semánticas como mecanismo de conexión entre las representaciones textual y formal del patrón (véase la sección 5.3), manteniendo así la esencia del patrón a la vez que se proporciona el fundamento necesario para la construcción de repositorios estandarizados y herramientas de soporte al proceso de automatización, como se muestra en la sección 6.3 del siguiente capítulo y en el que se describen dos herramientas, un repositorio web para la exploración de patrones y una herramienta CASE que integra los patrones en el proceso de diseño, las cuales comparten una misma representación y pueden utilizarse de manera complementaria.

5.1. Un modelo de representación basado en el conocimiento para pdh

Antes de comenzar con la descripción en sí del modelo de representación de patrones de diseño hipermedia, se desea justificar por qué es necesario optar por un nuevo mecanismo de representación. Como fue descrito en la sección 2.1.7 “Formalización de patrones”, todos los trabajos elaborados para llevar a cabo la transcripción de los patrones de diseño a un formato computacional han utilizado aproximaciones basadas en una **representación software**, siendo su objetivo final la incorporación codificada del patrón, es decir su solución en algún lenguaje de programación orientado a objetos, pero olvidando representar el resto de campos que dan valor al concepto de patrón. Dichas representaciones no soportan una mera exploración narrativa de los patrones, como por ejemplo recuperar aquellos patrones cuya intención fuese ayudar al usuario a orientarse dentro de la estructura de la aplicación (patrones `[BN1]Bread Crumbs` y `[BN2]Site Map`).

Además, hay que recordar que los patrones de diseño hipermedia se caracterizan por la heterogeneidad en su nivel de descripción que hace que, por un lado, algunos patrones no puedan ser aplicados de manera automatizada por encontrarse en un nivel de abstracción cercano a la manera que el usuario final visualiza la aplicación hipermedia, y que, por otro lado, aquellos patrones que sí son susceptibles de ser automatizados deben serlo en los diferentes métodos recogidos en la sección 2.2.3 “Métodos de desarrollo” los cuales no

comparten el mismo modelo de referencia para especificar la solución del patrón mediante entidades de diseño. Por lo tanto, es necesario un mecanismo de formalización flexible que permita tanto la interoperabilidad entre el conocimiento que captura el patrón y las diferentes representaciones de diseño finales, como una mera exploración dirigida para aquellos patrones que no puedan ser formalizados para su aplicación automática.

La conceptualización que aquí se propone aboga por una perspectiva basada en la **representación del conocimiento** independiente de las necesidades específicas de una determinada aplicación software y que permita representar cosas como la intención o la motivación del patrón, que son ideas abstractas que tienen cabida en el universo del discurso pero que no tienen una representación estructural o funcional. Además, se pretende enriquecer la representación textual del patrón con información semántica haciendo que aquellos patrones cuya aplicación no sea susceptible de ser automatizada en herramientas de desarrollo software puedan ser también accesibles por diferentes procesos software con objetivos como la exploración o la búsqueda de patrones.

Todas estas cuestiones relacionadas con la representación del conocimiento están siendo resueltas en la ingeniería del conocimiento, la inteligencia artificial y la ciencia de la computación mediante el uso de ontologías. En concreto, las ontologías están jugando en la actualidad un rol muy importante en el dominio de la web para el desarrollo de la web semántica [Berners-Lee *et al.*, 2001].

En el campo de la inteligencia artificial las ontologías surgieron para reutilizar y compartir conocimiento, de igual manera que surgieron los patrones en el campo de la ingeniería del software. El término **reutilizar** tiene como significado la construcción de nuevas aplicaciones ensamblando componentes ya creados, mientras que el término **compartir** denota la cualidad de que diferentes aplicaciones usen los mismos recursos. Las ontologías son la estructura básica o el armazón sobre el cual una base de conocimiento puede ser construida, libre de requerimientos técnicos.

El modelo de representación que se expondrá en las siguientes secciones de este capítulo es fruto de la aplicación de los principios de la ingeniería ontológica, dando como resultado una ontología del dominio de propósito específico que permite modelar los pdh a partir de un vocabulario descriptivo con una sintaxis y semántica apropiada.

Antes de pasar a desarrollar la arquitectura del modelo de representación para los patrones de diseño hipermedia y los componentes que la conforman en la sección 5.2, se

presentará brevemente diferentes conceptos del dominio de las ontologías dirigidos tanto a poner de manifiesto las cualidades que han llevado a la elección de este mecanismo como base de la representación propuesta, como a facilitar la interpretación del propio modelo de representación.

5.1.1. Ontologías

El término ingeniería ontológica hace referencia al conjunto de actividades que guían el diseño conceptual, implementación y desarrollo de ontologías [Devedzic, 2002].

La palabra ontología fue adoptada en el campo de la informática por [Neches *et al.*, 1991] del campo de la filosofía. Desde entonces se han dado muchas definiciones de lo que se entiende por ontología. En esta sección no se discutirán dichas definiciones (para ello se puede consultar [Guarino, 1997]), sino que se tomará la asumida por Gruber, y la consiguiente extensión realizada por Borst, que es la aceptada mayoritariamente:

> *"explicit specification of a conceptualization"*.
>
> [Gruber, 1993]

> *" formal specification of a shared conceptualization"*.
>
> [Borst *et al.*, 1997]

Estas dos definiciones fueron fusionadas en [Studer *et al.*, 1998] y explicadas en los siguientes términos: Una **conceptualización** es un modelo abstracto de algún fenómeno del mundo del cual se identifican los conceptos relevantes. **Explícito** significa que el tipo de conceptos usados y las restricciones sobre su uso están explícitamente definidos. **Formal** se refiere al hecho de que la ontología debería ser entendible por máquinas. Y **compartida** refleja la opinión de que una ontología captura un conocimiento consensuado, es decir, que no es privado, sino aceptado por un grupo o comunidad.

Las ontologías pueden desempeñar varios papeles o roles bajo el concepto unificador de compartir conocimiento [Gruber, 1993]:

- Repositorios para la organización de conocimientos e información, tanto de tipo corporativo como científico.

- Herramienta para la adquisición de información en situaciones en la que un equipo de trabajo la utiliza como soporte común para la organización del dominio.
- Herramienta de referencia en la construcción de sistemas basados en el conocimiento, ya que la utilización consistente de los términos que supone es básica en la ingeniería del conocimiento.
- Para permitir la reutilización del conocimiento ya existente en la creación de nuevas aplicaciones.
- Para permitir la indización, recuperación y divulgación de la información depositada en repositorios.

Las ontologías pueden ser modeladas con diferentes técnicas, entre las que destacan aquellas que combinan marcos con lógica de primer orden o lógica de descripción. Los diferentes componentes y términos que se pueden encontrar en una ontología son [Gómez-Pérez *et al.*, 2004]:

Clases: representan conceptos en su sentido más amplio. Pueden representar conceptos abstractos (intenciones, creencias, sentimientos, etc.) o conceptos específicos (personas, ordenadores, mesas, etc.). Por ejemplo, en un dominio de viajes podemos encontrar los conceptos Localización (ciudades, pueblos, etc.) y Medios de transporte (aviones, trenes, coches, autobuses, etc.). Las clases se pueden organizar en taxonomías a través de mecanismos de herencia.

Relaciones: representan un tipo de asociación entre conceptos del dominio. Las ontologías normalmente contienen relaciones binarias, en las que el primer argumento es conocido como el dominio de la relación y el segundo argumento es el rango. Por ejemplo, la relación *llegadaLugar* expresa que el punto de llegada de un Viaje (dominio) es una Localizacion(rango). Las relaciones binarias se pueden utilizar para representar **atributos** del concepto, también conocidos como propiedades o *slots*, que se distinguen porque su rango es un tipo de datos, como un número, una cadena de caracteres, etc. Por ejemplo, el atributo *numeroVuelo* define un código representado mediante un *string*. En lógica de descripción las relaciones son conocidas como roles.

Axiomas: representan aquellos fragmentos de conocimiento que no pueden ser implementados en la ontología con el resto de componentes del modelado y que son siempre

verdad. Además, son utilizados para verificar la consistencia de la propia ontología y la consistencia del conocimiento almacenado. Por ejemplo, un axioma en el dominio de viajes sería que no es posible viajar de EE.UU. a Europa en tren.

Instancias: representan elementos o individuos en una ontología. Por ejemplo, una instancia del concepto Localización puede ser Madrid.

Finalmente, las ontologías junto con un conjunto de instancias individuales de clases, forman lo que se conoce como base de conocimiento (*knowledge base*).

Existe una amplia variedad de lenguajes utilizados para formalizar ontologías. Algunos están basados en lógica de primer orden como KIF (Knowledge Interchange Format) [23], otros están especializados en recursos web como OIL (*Ontology Interchange Language*) [Fensel *et al.*, 2000], o en agentes como DAML (*DARPA Agent Markup Language*) [Hendler y McGuinness, 2000]. Estos dos últimos lenguajes se han unido en DAML+OIL [Mcguinness *et al.*, 2002] como el lenguaje para la web semántica, el cual está basado en XML y construido sobre RDF (*Resource Description Framework*) y su esquema (RDFS) [28]. Además, combina la sintaxis basada en marcos con la lógica de descripción, facilitando servicios de razonamiento. Este lenguaje ha servido como base para el estándar OWL (*Web Ontology Language*) del W3C (*World Wide Web Consortium*) [24]. OWL ofrece tres sublenguajes de expresividad incremental para satisfacer a cualquier usuario dependiendo de las características que necesite: OWL *Lite* para definir una clasificación jerárquica y algunas restricciones simples; OWL *DL* para conseguir la máxima expresividad, además de totalidad computacional y decisión; y OWL *Full* para alcanzar máxima expresividad, pero sin garantía de resolución computacional.

5.2. Arquitectura del modelo de representación

En la ingeniería ontológica el conocimiento es investigado en términos de los orígenes y elementos a partir de los cuales se produce, conocido esto como el universo del discurso. En este caso, el universo del discurso que se quiere formalizar es el de los patrones de diseño hipermedia, los cuales describen cómo solucionar aquellos problemas comunes que un diseñador puede encontrar cuando se enfrenta al desarrollo de un sistema hipermedia. Así, es necesario combinar el conocimiento de dos dominios diferentes: el dominio que

define **cómo se describe un patrón**, es decir, su formato, y el dominio implicado en **qué se describe en el patrón**, es decir, los componentes de un sistema hipermedia.

Hasta el momento ambos dominios no cuentan con una terminología común, y aunque una ontología debería representar el conocimiento consensuado de los expertos en el dominio, en este caso se ha optado por una aproximación basada en la literatura que permita desarrollar una propuesta inicial tal y como se propone en [Sicilia *et al.*, 2003], en la que los conceptos y las relaciones del dominio son obtenidos a partir de los recursos bibliográficos que los definen o tratan con ellos, permitiendo así un conocimiento consensuado aunque no explícitamente. Por lo tanto, el proceso de desarrollo que se ha seguido es un proceso iterativo basado en la literatura existente en ambos dominios, dando como resultado un modelo de representación semántico, cuya arquitectura puede verse en la figura 5.1. Este modelo de representación está formado por tres componentes:

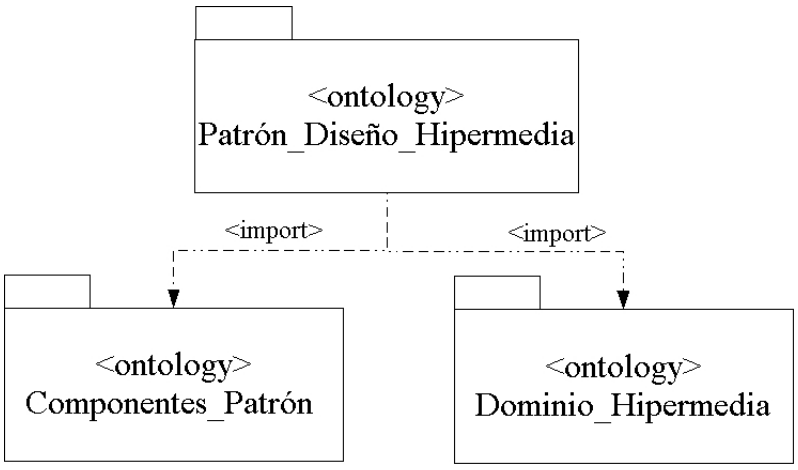


Figura 5.1: Modelo de representación

- Los **componentes del patrón** representan el formato a seguir para describir un patrón (v.g. nombre, motivación, problema, solución, etc.). Puesto que estos componentes son independientes del dominio al que pertenece el patrón, pueden utilizarse o refinarse para guiar la descripción de patrones pertenecientes a otros dominios.
- El **dominio hipermedia** representa el dominio sobre el cual se capturan los problemas y soluciones, es decir, aquellos conceptos y relaciones que describen las características generales de una aplicación hipermedia y que son utilizados para capturar la solución a un problema de diseño recurrente. En realidad, esta representación no

está asociada al dominio de los patrones sino al diseño de aplicaciones hipermedia, pudiendo ser utilizada y ampliada para otros objetivos como el modelado de aplicaciones hipermedia y la validación de meta-modelos [Montero *et al.*, 2003c].

- El **patrón de diseño hipermedia** representa explícitamente cómo se han de combinar los elementos de cada una de las representaciones anteriores dando lugar al armazón subyacente a partir del cual los patrones de diseño hipermedia pueden ser formalizados.

En las siguientes subsecciones se profundizará en cada una de estas representaciones formalizadas mediante el uso de ontologías las cuales serán descritas utilizando dos niveles de representación: una representación semiformal basada en marcos que permite describir gráficamente las clases, las propiedades y las relaciones entre clases (los nombres serán descritos en el idioma inglés) utilizando la herramienta EzOWL [16]; y una representación formal como es la lógica de descripción o descripciones (DL *Description Logic*), en concreto la rama denominada *SHOIN(D)* [Horrocks y Patel-Schneider, 2003] que proporciona un formalismo riguroso para inferir conocimiento a partir de la información existente, mediante la definición de conceptos, roles y axiomas. Ambas representaciones del conocimiento son subyacentes al fundamento del lenguaje de ontología para web OWL DL [24], el cual será utilizado como lenguaje de implementación final de las ontologías aquí presentadas (véase el apéndice B para un mayor detalle sobre la correspondencia entre OWL y DL). Aunque no se puede ignorar que UML es un estándar de facto y su uso está ampliamente difundido, no se ha utilizado como lenguaje de representación en este trabajo ya que una ontología necesita de un lenguaje con semántica formal para ser descrita, semántica de la que carece actualmente UML.

5.2.1. Ontología para los componentes de un patrón

Un patrón de diseño consta de una serie de elementos que le aportan una estructura uniforme, haciendo que sea más fácil de aprender, comparar y usar. Como ya se mencionó en las secciones 2.1 “Patrones de diseño en la ingeniería del software” y 2.1.5 “Patrones específicos del dominio”, muchos son los formatos que utilizan los autores de patrones para transmitir sus experiencias, aunque naturalmente, todos comparten las enseñanzas de Alexander. Por lo tanto, en esta conceptualización del formato de un patrón se tomará como

fuente literaria las definiciones y descripciones dadas por el propio Alexander, añadiéndose en la ontología del patrón de diseño hipermedia aquellos componentes específicos del dominio de aplicación (véase la sección 5.2.3).

Alexander desarrolló un formato de patrón para comunicar el conocimiento acumulado a través de la estructura que recoge el patrón, a la vez que permite compartir ese conocimiento y la naturaleza de mejoras que deben ser hechas. Esa plantilla fue descrita por el propio Alexander de la siguiente manera:

> "*First, there is a [pattern number, a title and a] picture, which shows an archetypal example of [a] pattern. Second, after the picture, each pattern has an introductory paragraph, which sets the context for the pattern, by explaining how it helps to complete certain larger patterns. Then there are three diamonds to mark the beginning of the problem. After the diamonds there is a headline, in bold type. This headline gives the essence of the problem in one or two sentences. After the headline comes the body of the problem. This is the longest section. It describes the empirical background of the pattern, the evidence for its validity, the range of different ways the pattern can be manifested in a building, and so on. Then, again in bold type, like the headline, is the solution - the heart of the pattern - which describes the field of physical and social relationships which are required to solve the stated problem, in the stated context. This solution is always stated in the form of an instruction - so that you know exactly what you need to do, to build the pattern. Then, after the solution, there is a diagram, which shows the solution in the form of a diagram, with labels to indicate its main components. After the diagram, another three diamonds, to show that the main body of the pattern is finished. And finally, after the diamonds there is a paragraph which ties the pattern to all those smaller patterns in the language which are need to complete this pattern, to embellish it, to fill it out.*"
>
> ([Alexander *et al.*, 1977], pp. x-xi)

De esta descripción se obtienen los siguientes componentes de un patrón:

Nombre: Nombra y enumera el patrón de manera inequívoca (*name*).

Imagen: Muestra un ejemplo del patrón (*image*).

Contexto: Introduce el contexto para el patrón, explicando cómo va a ayudar a completar ciertos patrones de mayor escala y hace referencia a aquellos patrones que deberían haber sido aplicados anteriormente (*context*).

Problema: Describe en una o dos frases la esencia del problema (*problem*).

Discusión: Describe los antecedentes empíricos del patrón, la evidencia para su validez y el rango de los diferentes modos en que el patrón se puede manifestar (*discussion*).

Solución: Describe el campo de las relaciones físicas y sociales requeridas para resolver el problema indicado, en el contexto indicado (*solution*).

Diagrama: Muestra la solución en forma de diagrama, con etiquetas que indican sus principales componentes (*diagram*).

Relacionado con: Enlaza el patrón a aquellos patrones de menor escala que son necesarios para completar a este patrón, adornarlo o rellenarlo (*relatedTo*).

La figura 5.2 recoge el concepto de **patrón de diseño** como una clase Pattern descrita a partir de una serie de atributos y relaciones que se identifican con los elementos anteriormente enumerados. Las relaciones *context* y *relatedTo* asocian instancias de la clase Pattern con otras instancias de la misma clase estableciendo así relaciones entre los propios patrones que permitirán describir el lenguaje de patrones. Las relaciones *problem* y *solution* establecen asociaciones entre la clase Pattern y la clase Actor. La clase Actor representa a aquellos elementos del dominio que participan en la descripción del estado del problema y de la solución del patrón y que van a ayudar a resolver dónde y cómo aplicar un patrón en un determinado diseño. En este momento el dominio del patrón no es conocido por lo que esta clase se ha definido de manera abstracta y se resolverá cuando se defina el patrón de diseño hipermedia en la sección 5.2.3. Este hecho posibilita reutilizar esta descripción semántica del patrón en cualquier otro dominio.

La representación en DL del concepto Pattern se muestra a continuación con sus cuantificadores y sus restricciones:

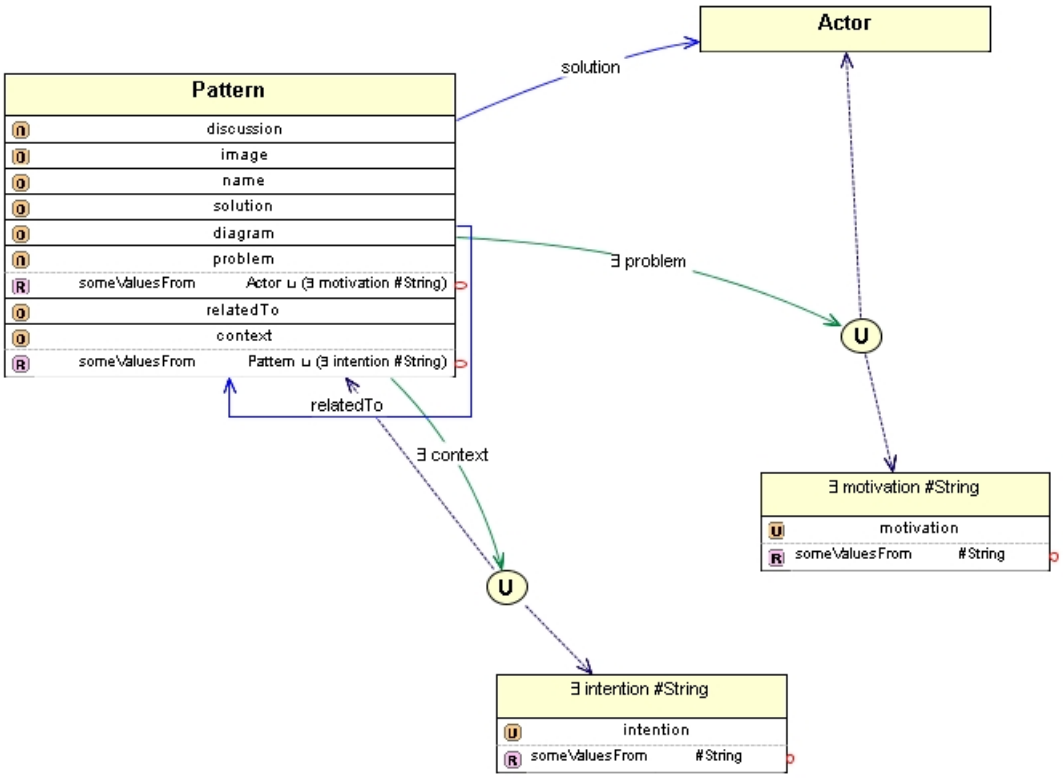


Figura 5.2: Ontología de los componentes de un patrón

$$
\begin{aligned}
Pattern \doteq \\
&\exists\, name.String \sqcap \exists\, image.URI \\
&\sqcap \exists\, context.(Pattern \sqcup (\exists\, intention.String)) \\
&\sqcap \exists\, problem.(Actor \sqcup (\exists\, motivation.String)) \\
&\sqcap \exists\, discussion.String \sqcap \forall\, solution.Actor \\
&\sqcap \exists\, diagram.URI \sqcap \forall\, relatedTo.Pattern
\end{aligned}
$$

Tanto para la propiedad *context* como para la propiedad *problem* se han incluido restricciones en el rango que permite recoger la descripción textual para la intención global del patrón (*intention*) y su motivación (*motivation*) respectivamente.

Sin embargo, se pueden realizar una serie de afirmaciones o axiomas sobre la clase Pattern y sus propiedades que posteriormente serán utilizadas por un sistema de inferencia para comprobar la integridad del conocimiento representado o nueva información:

- El nombre del patrón debe ser único para cada instancia de la clase Pattern y no puede haber dos instancias con el mismo nombre:

$$\top \sqsubseteq \leq\ 1\, name^{-}$$

- Un patrón fue definido por Alexander en su mínima expresión como:

> *Each pattern is a three-part rule which expresses a relation between a certain context, a problem, and a solution.* ([Alexander *et al.*, 1977])

Por lo tanto, cualquier instancia que tenga un contexto, un problema y una solución es un patrón.

$$(\exists = 1\, context \sqcap \exists = 1\, problem \sqcap \exists = 1\, solution) \sqsubseteq Pattern$$

- Para formar un lenguaje, el patrón debe formar parte de una red de patrones que trabajan en armonía para resolver un problema común de gran escala, es decir:

> *Each pattern then, depends both on the smaller patterns it contains, and on the larger patterns within which it is contained...*
>
> *...*
>
> *And it is the network of these connections between patterns which creates the language.*
>
> ([Alexander, 1979], pp. 312)

  Por lo tanto, las relaciones *relateTo* y *context* se definen como transitivas para poder inferir relaciones entre patrones de diferentes niveles:

$$relatedTo^{+} \sqsubseteq relatedTo$$

$$context^{+} \sqsubseteq context$$

- Una misma instancia no puede pertenecer a la clase Pattern y a la clase Actor:

$$Pattern \sqsubseteq \neg Actor$$

5.2.2. Ontología del dominio hipermedia

A partir del trabajo realizado para catalogar los patrones de diseño hipermedia (véase el capítulo 4), se ha detectado una falta de uniformidad a la hora de describir dichos patrones, bien porque el autor utiliza el vocabulario propio de un método de diseño concreto, o un vocabulario menos técnico y más cercano al usuario final de la aplicación. Por ejemplo, en los patrones descritos en [Garzotto *et al.*, 1999] se encuentran términos como *"Collection Center"* y *"Collection Members"*, es decir, un punto de acceso a un conjunto de páginas o nodos, que son definidos únicamente en el modelo *HDM* [Garzotto *et al.*, 1993], sin embargo en [van Duyne *et al.*, 2002] se pueden encontrar términos como *"Home"* y *"Elements"* para referirse a los términos anteriores y *"Browsing Hierarchy"* para la combinación de ambos. Este aspecto dificulta una de las principales motivaciones de los patrones: proporcionar un vocabulario común para expresar y comunicar su conocimiento.

Para que sean efectivos los catálogos de patrones deben ser presentados en un formato estándar usando una nomenclatura uniforme:

> *"Anyone who takes the trouble to consider it carefully can understand it and check against their own experience"*
> [Alexander, 1979]

En otras palabras, los patrones de diseño deberían ser igualmente accesibles para desarrolladores y usuarios, independientemente del método utilizado, proporcionando una lengua válida para el diálogo y la comunicación de problemas de diseño.

En [Garzotto *et al.*, 1999] se propone que los modelos de diseño hipermedia proporcionen el vocabulario necesario para la descripción de los patrones, sin que ejerzan influencia en la esencia misma de éstos. Su naturaleza es independiente del modelo concreto y se deberían poder especificar con las primitivas de cualquiera de ellos. Partiendo de esta premisa, y teniendo en cuenta que no existe ninguna ontología para el dominio de la hipermedia, se ha optado por estudiar los principales modelos de representación a partir de los cuales obtener un conjunto de términos y conceptos que permitan describir la estructura de una aplicación hipermedia. Esta ontología resultante se utilizará como representación común para capturar aquellos elementos hipermedia que forman parte del problema y de la solución del patrón de diseño, de tal forma que también se facilite la interoperabilidad con los diferentes modelos de diseño de cada uno de los métodos hipermedia en los que se podrán aplicar dichos patrones. También es necesario aclarar que esta ontología no pretende ser "la ontología del dominio hipermedia" sino resolver el problema de heterogeneidad en la descripción de los patrones descrito al comienzo de este apartado, a la vez que se cuentan con los términos hipermedia necesarios para capturar la esencia del patrón, por lo que tampoco se ha tenido en cuenta la total funcionalidad de los modelos a estudio.

En primer lugar, se estudiará el modelo de Dexter [Halasz y Schwartz, 1990], el cual proporciona una terminología de características propias de la hipermedia que describe su estructura, y que a partir de entonces se consideraría estándar en este campo. El objetivo de este modelo es proporcionar los principios básicos para comparar las características y funcionalidades de los sistemas de hipertexto, a la vez que sirve de base para desarrollar estándares de interoperabilidad e intercambio entre sistemas de hipertexto. Los principales elementos que se incorporarán a la ontología definidos en dicho modelo son:

- Un **componente** (Component) puede ser atómico, un enlace o una entidad compuesta formada de otros componentes. Cada componente tiene un identificador único (*uid*) y contiene un conjunto de atributos (*hasAttributes*) y especificaciones de presentación (*hasPresentationSpecifications*).

  - Los **atributos** (Attribute) pueden ser cualquier información relacionada con el componente y son descritas mediante el nombre de un atributo (*name*) y un valor (*value*). Por ejemplo, la fecha del último cambio del componente, el propietario del componente o el tipo del contenido del componente.
  - Las **especificaciones de presentación** (*hasSpecificationPresentation*)contiene la información de cómo el componente se presenta al usuario, pudiendo incluir cosas como el número de colores a mostrar o el tamaño o la localización de la ventana.

- Un componente atómico es lo que se conoce como **nodo** (Node).

- Un **enlace** (Link) representa una relación entre otros componentes. Para definir un enlace que va de una parte de una componente a otra parte de otro componente, es necesaria una entidad de direccionamiento indirecto llamada ancla.

- Un **ancla** (Anchor) tiene dos partes: un identificador (*anchorid*) y una localización (*location*) dentro de un componente.

- Un **componente compuesto** (CompositeComponent) está formado de otros componentes (*hasComponents*), atómicos o compuestos, que pueden ser tratados como un componente simple. Esta jerarquía de componentes está restringida a un grafo dirigido acíclico.

El modelo de Dexter, aunque según sus autores está orientado a hipermedia, se centra por completo en el hipertexto, dejando de lado el estudio de la variedad de tipos posibles, de carácter multimedia, entre los cuales puede haber dependencias de tiempo. Es por ello que posteriormente apareció el modelo de Amsterdam [Hardman *et al.*, 1994], que basado en Dexter, añade el estudio del tiempo, entre otras cosas. Las principales aportaciones, de Amsterdam, que serán incluidas a los elementos descritos anteriormente son:

- Dos tipos de mecanismo de composición:

  - Uno **temporal** que se define entre componentes (*hasSource* y *hasTarget*) y permite, por ejemplo, composiciones multimedia en paralelo y en secuencial. Estas relaciones temporales son especificadas como arcos de sincronización (Synchronization) seguidos por la restricción temporal entre los componentes (*type*) y los retardos permitidos (*begin end*).
  - Otro **espacial** que permite mostrar los contenidos de un componente según unas especificaciones de presentación.

- Un **enlace contextual** (ContextualLink) es un nuevo componente (típicamente compuesto) que contiene la colección de los componentes afectados por un enlace (*context*). El mecanismo de contexto permite definir unas opciones de visualización diferentes para cada enlace. En particular, puede ser que parte de la información continúe visible, mientras que otra deja de estarlo.

- Un **elemento raíz** (Root) que representa al componente inicial del cual parte el grafo acíclido dirigido que forman todos los componentes de una aplicación hipermedia.

Finalmente, el modelo Labyrinth [Díaz *et al.*, 1997] añade a los modelos anteriores, entre otras cosas, tener en cuenta al usuario como un elemento más del hiperdocumento, permitiendo así ofrecer un acceso seguro o personalizado a través de la definición de políticas de acceso. Desde este modelo se incorporan a la ontología los siguientes conceptos:

- El **contenido** (Content) pasa a ser un componente con identidad propia, separando así al contenedor abstracto que forman la estructura hipertextual (nodos) de los elementos de información.

- Un **nodo** es un contenedor de información (*hasContents*).

- Un **nodo compuesto** además de representar una relación de agregación, puede representar relaciones de generalización (*relationship*).

- Un **enlace** establece una conexión entre dos conjuntos de anclas, el origen (*hasSource*) y el destino (*hasTarget*), que representa los caminos de navegación del sistema hipermedia.

- Los contenidos, a parte de arcos de sincronización, pueden tener arcos de alineación (Alignment) que definen restricciones espaciales de presentación.

- Un **usuario** (User) es un representante de un tipo de usuario final o un grupo de usuarios tipo para tratar con hiperdocumentos personalizados y seguros. Una etiqueta tipo (*userType*) especifica si es un usuario tipo (*role*) o un grupo (*team*). Cada usuario tiene asociado también una lista de usuarios (*userList*).

- Una **categoría de acceso** (AccessCategory) define una categoría de seguridad (*securityCategory*) para un determinado usuario (*hasUser*) y un determinado nodo o contenido (*hasObject*).

- Las **categorías de seguridad** (*securityCategory*) asociadas a nodos y contenidos definen las operaciones permitidas a los usuarios, que se definen como un orden parcial:

  - *Browsing*, el nodo o contenido sólo puede ser visitado.
  - *Personalizing*, el nodo o contenido puede además, ser personalizado.
  - *Editing*, el nodo o contenido puede además, ser modificado.

- Un **evento** (Event) es definido como una condición (*condition*) cuyo cumplimiento causa una reacción (*action*).

En la figura 5.3 se muestra una ontología que representa los elementos hipermedia anteriormente descritos.

La representación en DL de los conceptos representados en la figura 5.3 junto con sus cuantificadores y sus restricciones se muestra a continuación:

$$
\begin{aligned}
Attribute &\doteq \exists name.String \sqcap \exists value.String \\
Component &\doteq \exists uid.String \sqcap \exists hasAttributes.Attribute \\
&\sqcap \exists hasSpecificationPresentation.Attribute \sqcap \exists hasEvents.Event
\end{aligned}
$$

$$
\begin{aligned}
CompositeComponent &\sqsubseteq Component \\
CompositeComponent &\doteq \exists hasComponents.(Node \sqcup CompositeComponent) \\
&\sqcap \exists relationship.(\{generalization, aggregation\}) \\
&\sqcap \exists securityCategory.(\{browsing, personalizing, editing\})
\end{aligned}
$$

$$Node \sqsubseteq Component$$
$$Node \doteq \exists hasConstraints.Constraint \sqcap \exists hasContents.Content \sqcap \exists hasAnchors.Anchor$$

$$Content \sqsubseteq Component$$
$$Content \doteq \exists securityCategory.(\{browsing, personalizing, editing\})$$

$$Link \sqsubseteq Component$$
$$Link \doteq \exists hasSource.Anchor \sqcap \exists hasTarget.Anchor$$

$$ContextualLink \sqsubseteq Link$$
$$ContextualLink \doteq \exists context.Component$$

$$User \sqsubseteq Component$$
$$User \doteq \exists userList.User \sqcap \exists userType.(\{role, team\})$$

$$Anchor \doteq \exists anchorID.String \sqcap \exists location.(Node \sqcup Content)$$

$$
\begin{aligned}
Constraint &\doteq \exists type.String \sqcap \exists hasSource.(Node \sqcup Content) \\
&\sqcap \exists hasTarget.(Node \sqcup Content) \\
Alignment &\sqsubseteq Constraint \\
Synchronization &\sqsubseteq Constraint \\
Synchronization &\doteq \exists begin.Integer \sqcap end.Integer
\end{aligned}
$$

$$
Event \doteq \exists condition.String \sqcap \exists action.String
$$

Además, una serie de afirmaciones (axiomas) se pueden realizar sobre el conjunto de clases y propiedades aquí presentados:

- Cada componente tiene un identificador único (*uid*):

$$\top \sqsubseteq \leq 1 uid^{-}$$

- Las subclases de la clase Component son excluyentes:

$$CompositeComponent \sqsubseteq \neg Content$$

$$CompositeComponent \sqsubseteq \neg Link$$

$$CompositeComponent \sqsubseteq \neg Node$$

$$CompositeComponent \sqsubseteq \neg User$$

$$User \sqsubseteq \neg Link$$

$$User \sqsubseteq \neg Content$$

$$User \sqsubseteq \neg Node$$

$$User \sqsubseteq \neg CompositeComponent$$

$$Link \sqsubseteq \neg Content$$

$$Link \sqsubseteq \neg Node$$

$$Content \sqsubseteq \neg Node$$

- Las subclases de la clase Constraint son excluyentes:

$$Aligment \sqsubseteq \neg Synchronization$$

- La relación que contiene la lista de usuarios es transitiva para poder inferir usuarios relacionados:

$$userList^{+} \sqsubseteq userList$$

- Un hiperdocumento (Hyperdocument) es equivalente a la unión de las instancias de la clase Attribute, la clase Component, la clase Constraint, la clase AccessCategory y la clase Event:

$$Hyperdocument \equiv Attribute \sqcup Component \sqcup Constraint \sqcup AccessCategory \sqcup Event$$

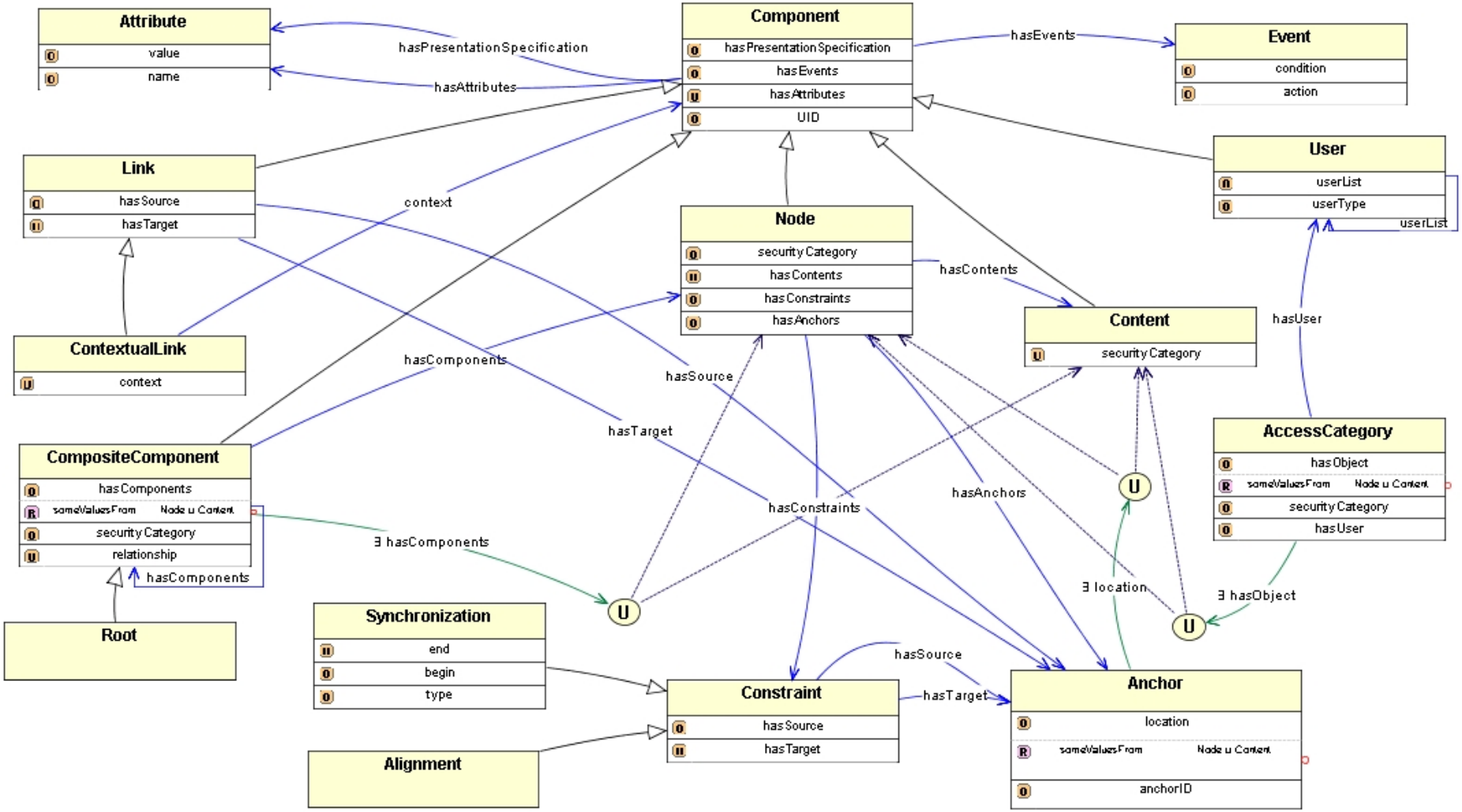


Figura 5.3: Ontología de la estructura para el diseño de una aplicación hipermedia

5.2.3. Ontología del patrón de diseño hipermedia

La representación formal del patrón de diseño hipermedia es producto de la combinación de las dos ontologías anteriormente descritas, la ontología de los componentes del patrón, la cual proporciona el formato a seguir a la hora de describir de manera literaria un patrón y la ontología del dominio hipermedia, la cual proporciona los elementos que permiten describir la situación del problema y la solución que plantea el patrón.

Como se mencionó en la sección 2.3.1, no existe un formato único de descripción para los patrones de diseño hipermedia. El formato que más se está utilizando en estos momentos es el de Alexander, debido sobre todo al éxito del libro *"Design of sites"* [van Duyne *et al.*, 2002] que utiliza dicho formato para describir sus patrones web. Partiendo de dicho formato, y que ha sido representado en la ontología descrita en la sección 5.2.1 por la clase Pattern se propone una especialización de dicha clase (HypermediaPattern) con dos nuevas propiedades con el objetivo de recoger la categorización de patrones propuesta en la sección 4.1 y poder clasificar así a los patrones según su:

- **Aspecto de diseño** (*aspect*), cuyos posibles valores son Navegación, Estructura, Presentación, Interacción, Personalización y Seguridad.

- **Nivel de descripción** (*level*), cuyos posibles valores son Alto, Medio y Bajo.

La propiedad *relatedTo* definida para la clase Pattern ha sido especializada para recoger los diferentes tipos de conexiones definidos en la sección 4.3, los cuales permiten la combinación de los patrones formando un lenguaje:

- Un patrón utiliza (*usesTo*) a otro patrón como parte de su solución.

$$usesTo \sqsubseteq relatedTo$$

- Un patrón sugiere (*suggestsTo*) un nuevo problema que es tratado por otros patrones.

$$suggestsTo \sqsubseteq relatedTo$$

- Un patrón sugiere a otro patrón que es una especialización (*specilizedBy*) del problema que aborda.

$$specilizedBy \sqsubseteq relatedTo$$

En la figura 5.4 queda reflejada la nueva ontología que representa a un patrón de diseño hipermedia.

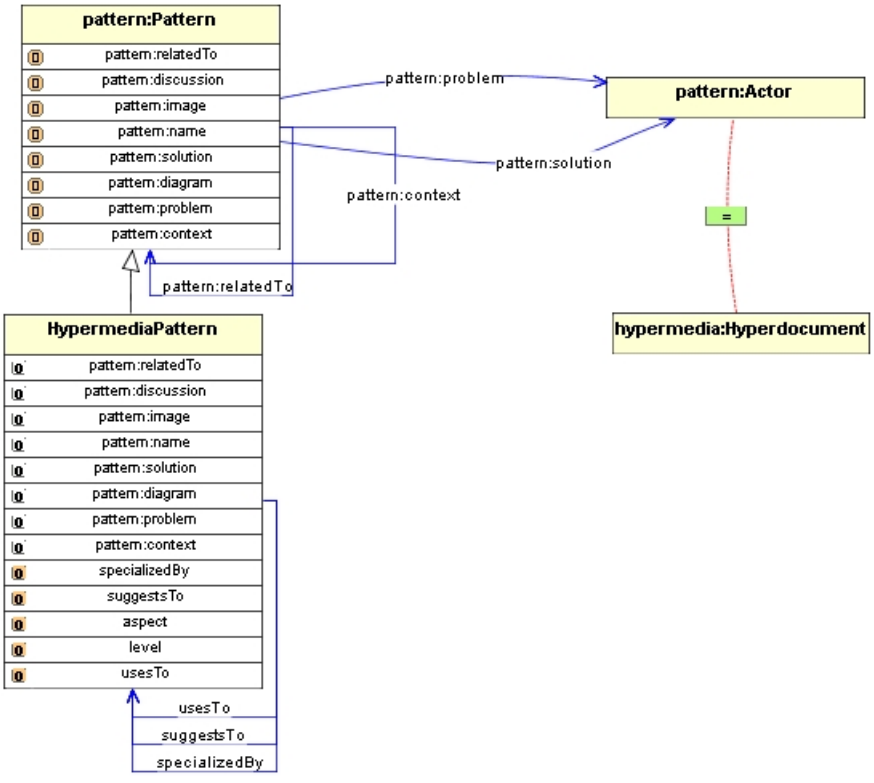


Figura 5.4: Ontología del patrón de diseño hipermedia

La representación en DL del concepto HypermediaPattern quedaría así:

$$HypermediaPattern \sqsubseteq Pattern$$
$$HypermediaPattern \doteq$$
$$\exists level.(\{High, Medium, Low\}) \sqcap$$
$$\exists aspect.(\{Navigation, Structure, Presentation, Personalization, Security, Interaction\}) \sqcap$$
$$\forall usesTo.HypermediaPattern \sqcap$$
$$\forall suggestTo.HypermediaPattern \sqcap$$
$$\forall specializedBy.HypermediaPattern$$

Además, la clase Actor, definida en la ontología de los componentes de un patrón para representar a cualquier elemento del dominio involucrado en la descripción del patrón, es equivalente en el dominio hipermedia a la clase Hyperdocument, la cual fue definida como la extensión de los elementos hipermedia que describen la aplicación.

$$Actor \equiv Hyperdocument$$

5.2.4. Una base de conocimiento de pdh

Los conceptos y las relaciones definidas en las secciones anteriores representan el conocimiento de cómo describir un patrón mediante los elementos del dominio al que pertenece, el dominio de las aplicaciones hipermedia, conformando la terminología, que en lógica de descripción recibe el nombre de TBox (*Terminological Box*), a partir de la cual se puede definir los individuos del dominio, los patrones de diseño hipermedia, conocido como ABox (*Assertional Box*).

Un ejemplo de instancia de la clase HypermediaPattern sería el patrón `[MN1]Index Navigation`, el cual aborda un problema de **navegación** de nivel **medio**. La intención de este patrón es **proporcionar acceso directo** a la información definida por el patrón `[ME1]Hierarchical Organization`. A partir de este contexto se plantea cómo **proporcionar acceso directo** a un determinado **miembro** de un **grupo de nodos**. Las razones de la utilidad de este patrón son que **el usuario no es capaz de hacer una elección**, que **no es capaz de identificar los elementos de la lista** y que **necesita volver a un punto de par-**

tida para acceder a otro nodo, teniendo que realizar así dos pasos en la navegación. Por lo tanto, la solución que se propone es definir un **punto de entrada** que contenga **enlaces de ida** a cada uno de los miembros del **grupo de nodos** y **enlaces de vuelta** desde cada uno de los miembros al punto de entrada. Además, para acelerar la navegación, se puede mantener accesible el punto de entrada.

A partir de lo aquí descrito se pueden realizar una serie de aserciones que representan formalmente la esencia de este patrón siguiendo la formalización propuesta en el apartado anterior.

```
HypermediaPattern(IndexNavigation)
name(IndexNavigation,"Index Navigation")
level("Medium")
aspect("Navigation")
context(IndexNavigation, HierarchicalOrganization)
context(IndexNavigation,"acceder información")
problem(IndexNavigation,"proporcionar acceso directo")
problem(IndexNavigation,GrupoNodos)
problem(IndexNavigation,MiembroNodo)
CompositeComponent(GrupoNodos)
Node(MiembroNodo)
hasComponents(GrupoNodos,MiembroNodo)
relationship(GrupoNodos,"specialization")
disscusion(IndexNavigation,"el usuario no es capaz de hacer una elección")
disscusion(IndexNavigation,"no ser capaz de identificar los elementos de la lista")
discussion(IndexNavigation,"no tener una meta específica")
discussion(IndexNavigation,"necesidad de volver a un punto de partida para acceder a otro nodo")
solution(IndexNavigation,EnlaceIda)
solution(IndexNavigation,PuntoEntrada)
Link(EnlaceIda)
Node(PuntoEntrada)
Anchor(PuntoEntradaAnchor)
location(PuntoEntradaAnchor,PuntoEntrada)
Anchor(GrupoNodosAnchor)
location(GrupoNodosAnchor,GrupoNodos)
hasSource(Enlace,GrupoNodosAnchor)
hasTarget(Enlace,PuntoEntradaAnchor)
solution(IndexNavigation,EnlaceVuelta)
hasSource(EnlaceVuelta,PuntoEntradaAnchor)
hasTarget(EnlaceVuelta,GrupoNodosAnchor)
solution(IndexNavigation,NodoAcceso)
CompositeComponent(NodoAcceso)
hasComponents(NodoAcceso,PuntoEntrada)
hasComponents(NodoAcceso,GrupoNodos)
relationship(NodoAcceso,"aggregation")
hasSpecificationPresentation(NodoAcceso,"frame")
diagram(IndexNavigation,"file://hypatterns.no-ip.info/repositorio/indexnavigationpattern.jpg")
usedTo(IndexNavigation,CollectionCenter)
suggestsTo(IndexNavigation,GuidedTour)
suggestsTo(IndexNavigation,LocationBreadCrumbs)
specilizedBy(IndexNavigation,NavigationalContext)
HypermediaPattern(HierarchicalOrganization)
HypermediaPattern(CollectionCenter)
HypermediaPattern(GuidedTour)
HypermediaPattern(LocationBreadCrumbs)
HypermediaPattern(NavigationalContext)
```

A partir de este ejemplo se puede ver cómo se combinan las dos terminologías para formalizar un patrón de diseño hipermedia: por un lado, el dominio de los componentes del patrón proporciona los campos de los que consta el patrón, mientras que el dominio hipermedia describe el problema y la solución que plantea dicho patrón. En el anexo A puede verse la descripción completa de este patrón y las aserciones aquí enumeradas, descritas en el lenguaje RDF, el cual ha sido utilizado para representar el Abox.

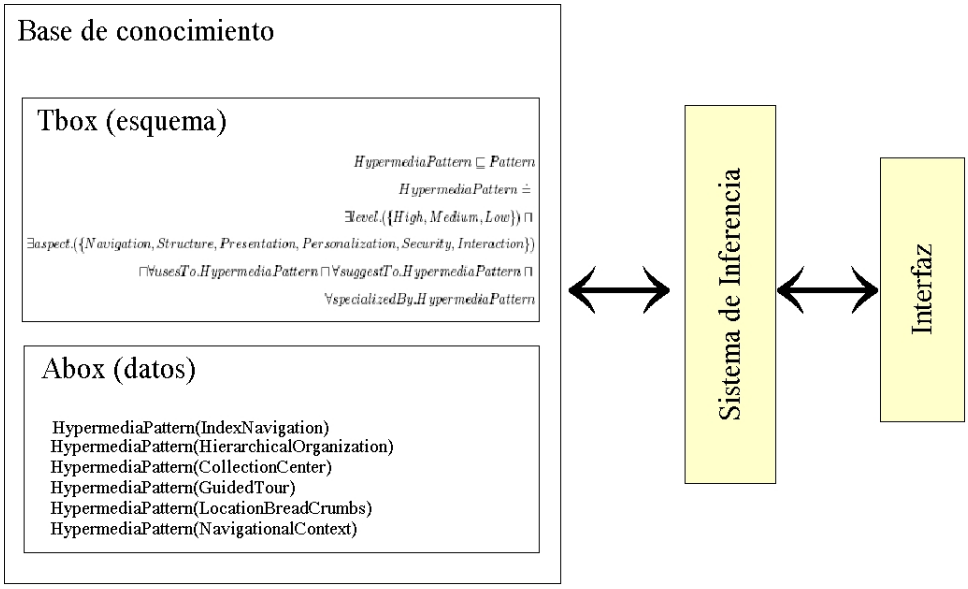


Figura 5.5: Una base de conocimiento de pdh

En la figura 5.5 se muestra como la representación formal de un patrón de diseño hipermedia (Tbox), y un conjunto de instancias (Abox) conformarían una base de conocimiento de patrones de diseño hipermedia a partir de la cual se puede inferir automáticamente el conocimiento implícito con la ayuda de un sistema de inferencia. Estos sistemas permiten comprobar si el conocimiento definido es correcto, verificar la pertenencia de una instancia a un determinado concepto, verificar que la construcción de instancias se hace de acuerdo a lo definido por la terminología, recuperar individuos mediante consultas, es decir, un conjunto de funciones que permiten la construcción de herramientas software basadas en el conocimiento, y en este caso, aplicaciones que permitan la gestión, la organización, la recuperación y la aplicación de aquellos patrones susceptibles de ser automatizados.

Esta representación no es suficiente para permitir el desarrollo de aplicaciones basadas en la exploración textual del patrón ya que su descripción no forma parte de la representa-

ción, aunque se puede utilizar como formalismo subyacente para completar la descripción textual de los patrones, como se verá en la siguiente sección.

5.3. Anotaciones semánticas para pdh

La información puede ser enriquecida de manera declarativa mediante **anotaciones** que ofrezcan una semántica interpretable para un ordenador, a partir de la cual se puedan escribir programas que permitan el acceso inteligente a dichos recursos. Estas anotaciones se pueden aplicar a cualquier tipo de texto, sonido, vídeo o, como actualmente está sucediendo, a páginas web para desarrollar la idea de la web semántica [Berners-Lee *et al.*, 2001]. En concreto, los recursos web serán anotados con metadatos que pueden ser aprovechados por programas software para realizar un acceso inteligente a dichos recursos, facilitar la búsqueda y la navegación en la web, y explotar nuevos enfoques de inferencia a partir de esos recursos, mientras que se sigue mantenido la presentación de la información para las personas.

Teniendo en cuenta esta aproximación y las investigaciones y resultados de este nuevo campo [30], se ha decidido utilizar una aproximación basada en anotaciones semánticas como medio para transformar la representación textual de un patrón en una estructura de conocimiento entrelazada que represente la información relevante subyacente. Como resultado del proceso de anotación se obtiene una semántica interpretable por un ordenador, pasando de un nivel sintáctico a un nivel semántico, a la vez que el usuario sigue teniendo accesible la descripción textual del patrón.

A continuación se realiza una breve introducción a los principales conceptos y términos relacionados con las anotaciones semánticas (véase la sección 5.3.1), para posteriormente describir cómo esta aproximación ha sido aplicada para anotar la descripción textual de los patrones (véase la sección 5.3.2) a partir de los conceptos definidos en el modelo de representación en la sección 5.2. Finalmente, se presenta una herramienta que ha sido desarrollada *ex profeso* como soporte al proceso de anotación y el desarrollo de repositorios semánticos de patrones de diseño hipermedia (véase la sección 5.3.3).

5.3.1. Anotaciones semánticas basadas en ontologías

La anotación es invisible para el humano, pero las máquinas pueden leer y entender ese contenido. **Anotar semánticamente** un documento significa asignar a las entidades del texto enlaces a sus descripciones semánticas. Dichas descripciones semánticas pueden ser obtenidas a partir de una ontología que proporcionará tanto información de la clase como de la instancia de las entidades.

Una anotación semántica se formaliza a partir de dos tipos de elementos, el documento y la representación formal, y dos funciones, una función desde el documento a la representación formal, llamada anotación y una función desde la representación formal al documento llamada índice [Euzenat, 2002]. En la figura 5.6 queda representada una anotación realizada sobre texto regular que representa a un patrón y como es anotado a su representación semántica a través de sus instancias. Por ejemplo, la descripción textual que representa al nombre del patrón (`[MN1]Index Navigation`) ha sido anotada mediante la creación de una instancia del tipo *name* que forma parte del concepto HypermediaPattern.

Por lo tanto, los elementos necesarios para la representación de anotaciones semánticas son:

- Una ontología que defina los conceptos y las relaciones del dominio a anotar.
- Los identificadores de los conceptos que serán enlazados a su descripción semántica (creación de instancias).
- Una base de conocimiento con las descripciones de los conceptos.

5.3.2. El proceso de anotación de los pdh

Teniendo en cuenta el concepto de anotación semántica presentado en la sección previa, las anotaciones se utilizarán para transformar recursos sintácticos existentes, en este caso, la descripción textual del patrón, en estructuras de conocimiento entrelazadas que representan información subyacente importante, como pueda ser el nombre del patrón, con qué otros patrones está relacionado, qué elementos de diseño forman parte de la solución que propone el patrón, etc.

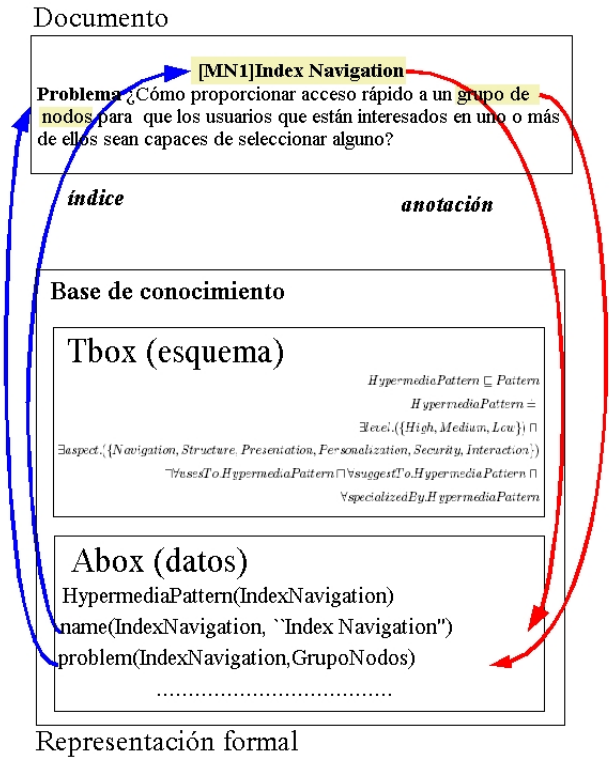


Figura 5.6: Anotación semántica

Para llevar a cabo el proceso de anotación, es necesario previamente desarrollar la estructura semántica que servirá de base a dicho proceso. En la sección 5.2.3 se definió una ontología que describe el concepto de patrón de diseño hipermedia y sus relaciones, la cual proporcionará al anotador el conocimiento necesario sobre qué elementos ontológicos tiene que usar en cada momento, es decir, si se trata de una instancia, de una propiedad, de una clase o la selección de un valor predefinido. Además, debido a cómo se definió la estructura de la ontología de un patrón hipermedia, el anotador tendrá conocimiento del dominio sobre:

- los componentes de un patrón y con qué tipo de términos pueden ser anotados, a partir de los cuales se definirá el armazón de un patrón de diseño (véase la sección 5.2.1 “Ontología para los componentes de un patrón”);

- y qué elementos son necesarios para anotar aquellos campos del patrón que describen características de las aplicaciones hipermedia (véase la sección 5.2.2 “Ontología del dominio hipermedia”).

La figura 5.7 muestra el proceso de anotación. Primeramente, el anotador debe identificar aquella palabra o palabras claves que definen la esencia de cada uno de los componentes del patrón y asignarle el concepto o la relación correspondiente según se describió en la ontología (1). Por ejemplo, para el campo *problem* se definió como una relación entre una instancia de la clase Pattern y la combinación de la intención del patrón y los elementos hipermedia que presentan el estado inicial del problema del patrón. En este punto el anotador debe decir qué tipo de instancia crear y asignarle el valor del texto identificado.

Para mantener la asociación entre el texto y la instancia creada es necesario un localizador, es decir, una anotación (2). Al finalizar el proceso de anotación, se extrae el patrón formalizado y se almacena de manera separada con la descripción textual y las anotaciones, para su posterior procesamiento (3).

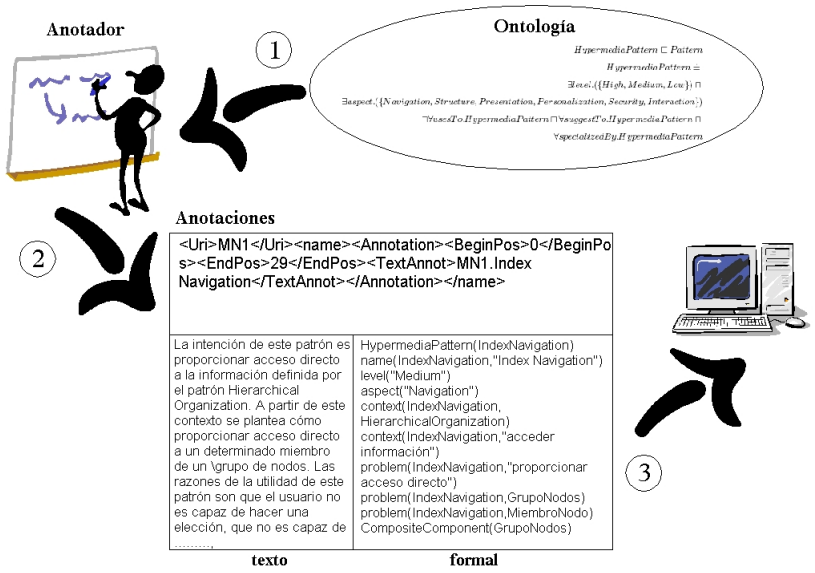


Figura 5.7: El proceso de anotación de un pdh

Esta combinación de representación textual con su representación formal tiene como objeto el desarrollo de repositorios de patrones de diseño hipermedia que puedan ser compartidos por herramientas software para la exploración y la aplicación de dichos patrones,

como se ilustrará en el capítulo 6 “Evaluación”. Para dar soporte al anotador durante el proceso de anotación y a la creación de repositorios para pdh se ha desarrollado una herramienta que se presenta en la siguiente sección.

5.3.3. AnnotPat: una herramienta de anotación para pdh

Expresar formalmente un patrón utilizando ontologías como la presentada en este capítulo puede resultar difícil para la mayoría de los usuarios de los patrones, pues no suelen ser ingenieros del conocimiento cualificados. Los usuarios son capaces de entender e interpretar la descripción textual del patrón e identificar los términos propios de su dominio, representados mediante una ontología, pero no de describir un patrón concreto utilizando un lenguaje para la descripción de recursos como RDF y crear así sus propios repositorios.

Ante la necesidad de dotar al usuario de los pdh de un mecanismo que le permita formalizar sus propios patrones sin un incremento excesivo en la carga cognitiva que ya supone entenderlos y utilizarlos, como ya ha quedado plasmado a lo largo de este trabajo, son necesarias herramientas de soporte al proceso de anotación.

Aunque en la actualidad existen diversas herramientas que realizan la anotación de documentos de manera automática [3], las anotaciones manuales son más precisas y, lo que es más importante, el usuario puede hacer su propio análisis del documento identificando, según su criterio, cuáles son los elementos que mejor recogen la esencia del patrón. Para ayudar a los usuarios a crear anotaciones correctas que capturen su conocimiento se ha desarrollado una herramienta de anotación, llamada AnnotPat (Annotation Pattern) [Montero *et al.*, 2004b], que proporciona como funcionalidad principal el soporte a la anotación de la descripción textual de un patrón con los términos semánticos apropiados.

La figura 5.8 muestra la interfaz de la herramienta de anotación que se encuentra dividida en dos partes. El lado izquierdo muestra el estado de las anotaciones hechas por el usuario como una instancia de la ontología del patrón de diseño hipermedia (1) y de la ontología del dominio hipermedia (2). El lado derecho muestra la plantilla del patrón (3) extraída a partir de la primera ontología, y donde la escritura del patrón y sus anotaciones pueden ser realizadas utilizando el vocabulario definido en las ontologías. La herramienta está soportada por una base de conocimiento donde se almacenan tanto las ontologías que permiten guiar el proceso de anotación, como las anotaciones generadas a partir de dicho

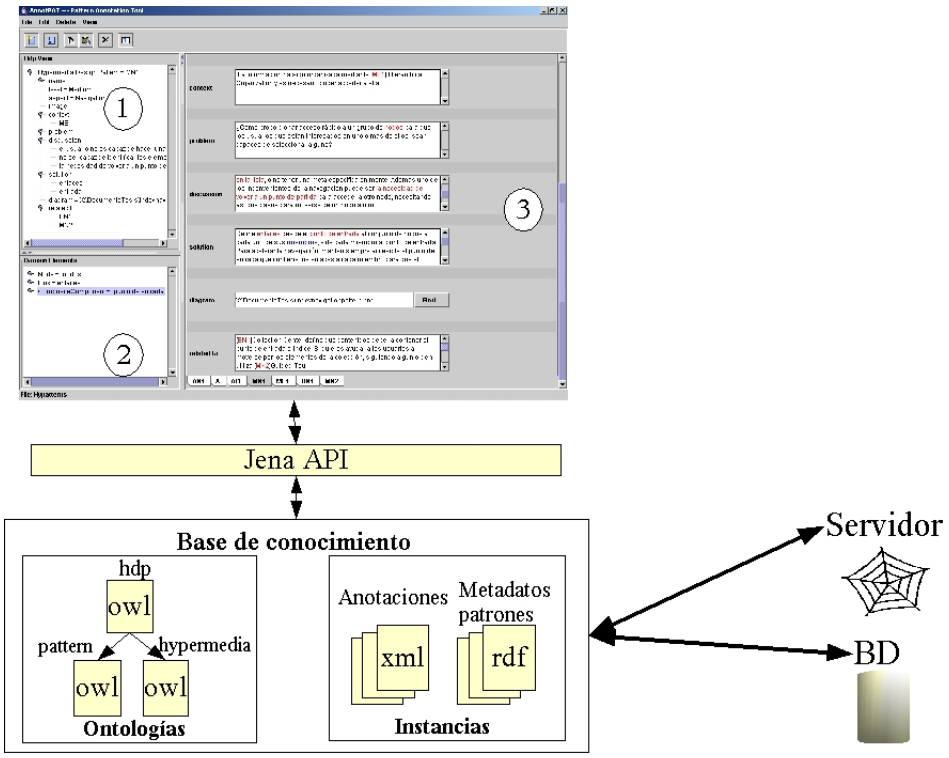


Figura 5.8: Arquitectura de la herramienta de anotación AnnotPat

proceso. La base de conocimiento puede ser almacenada en local, en una base de datos o en un servidor web. Este último caso será tratado en el siguiente capítulo en la sección 6.3.1.

En cuanto a cuestiones técnicas la herramienta se ha desarrollado en Java en su versión 1.4. Como ya se ha mencionado, las ontologías están expresadas en el lenguaje OWL y recogidas en el anexo B, mientras que las anotaciones son almacenadas en XML y las instancias relacionadas con la ontología en un repositorio RDF. Para manejar estos tipos de ficheros e inferir conocimiento se ha utilizado la herramienta Jena [22].

AnnotPat trabaja con repositorios que pueden contener uno o varios patrones y permite trabajar simultáneamente con ellos. Cada patrón es mostrado en una pestaña independiente cuya selección provoca la visualización tanto del contenido del patrón como de sus anotaciones. La plantilla del patrón está formada por una serie de campos que se adaptan al tipo de rango definido para los atributos de la clase HypermediaPattern, campos de selección o de texto libre. En el primer caso, la mera selección del elemento lleva asociada la creación de una anotación con dicho valor. En el segundo caso, el proceso de anotación comienza

seleccionando una palabra o grupo de palabras a las que se quiera asociar su representación semántica y sobre ella al pulsar el botón izquierdo del ratón, una ventana emergente mostrará al usuario los posibles elementos con que puede anotar el texto según el campo del patrón donde se encuentre. Para ayudar a diferenciar las palabras que ya han sido anotadas, el texto aparecerá en color rojo (véase la figura 5.9).

Tras la creación de una anotación, se actualizará el árbol de anotaciones con una nueva entrada. Las anotaciones también pueden ser eliminadas posicionando el cursor sobre la anotación y pulsando el botón izquierdo del ratón la misma ventana emergente mostrará la entrada "Remove Annotation" habilitada. La realización de está acción llevará consigo la actualización del árbol de anotaciones.

La figura 5.9 muestra la pantalla capturada tras realizar el proceso de anotación sobre el patrón `[MN1]Index Navigation` que podría ser realizado por el autor del patrón o por un diseñador hipermedia.

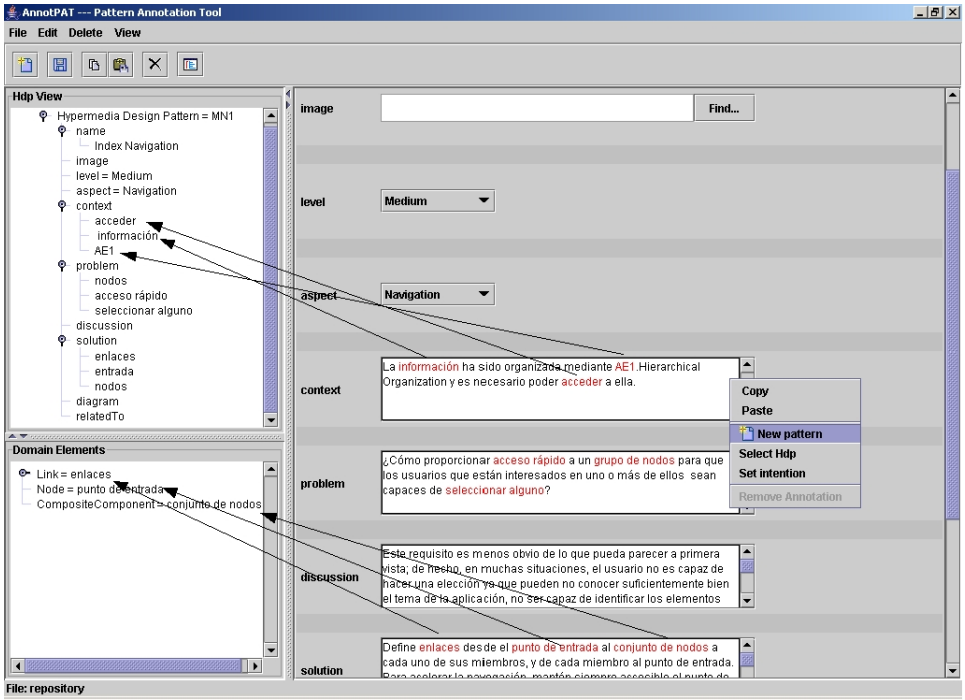


Figura 5.9: Un patrón anotado semánticamente con AnnotPat

5.4. Resumen del capítulo

Este capítulo parte de la situación actual a la que se enfrenta un diseñador hipermedia a la hora de utilizar los patrones, independientemente del catálogo y del lenguaje presentados en el capítulo anterior. Los pdh existentes, unos 200, se encuentran repartidos entre diferentes publicaciones, algunos de ellos accesibles vía web y otros no. En cualquier caso es el usuario el que debe recorrer cada uno de los recursos, intentar recuperar aquellos patrones relevantes basándose bien en el nombre del patrón o en los descriptores utilizados para su clasificación, decidir si es el adecuado, interpretar y adaptar la solución planteada y finalmente, comprobar si con ello ha satisfecho la necesidad que originó la utilización de patrones.

Esta situación hace que los dos únicos condicionantes que determinan el éxito o el fracaso del uso de los patrones sean el conocimiento que pueda tener el diseñador sobre la existencia de estos recursos y su entendimiento sobre cuándo debería ser aplicado un patrón. Sin embargo, este contexto podría ser aliviado con el desarrollo de repositorios que permitan la gestión, la organización y la recuperación de patrones a partir de los cuales construir herramientas software para la exploración y aplicación de los patrones.

En este capítulo se ha presentado una representación formal basada en ontologías para los pdh. Esta representación combina el conocimiento de dos dominios diferentes: el dominio que define cómo se describe un patrón, su formato, y el dominio implicado en que se utiliza en el patrón, es decir, el de los componentes para el diseño de un sistema hipermedia.

La utilización de ontologías ha aportado a esta representación un formalismo riguroso, como es la lógica de descripción, que permite el desarrollo de bases de conocimiento para pdh, así como comprobar la consistencia del conocimiento descrito. Además, desde que las ontologías están jugando un papel importante en el desarrollo de la web semántica para el procesamiento automático de los recursos web mediante su asociación a sus descripciones semánticas, estas investigaciones también se han tenido en cuenta para la formalización de los pdh. En concreto el modelo de representación definido se ha utilizado como conocimiento subyacente para anotar la descripción textual del patrón con su correspondiente descripción semántica. Esta unión permite tener una representación dual del patrón que puede ser compartida y reutilizada entre personas y sistemas computacionales.

Para finalizar este capítulo, se ha presentado una herramienta llamada AnnotPat, que da soporte tanto al autor de patrones como al diseñador hipermedia durante el proceso de anotación, así como al desarrollo de repositorios.

En el siguiente capítulo se procederá a presentar la evaluación tanto de la propuesta para la organización del espacio de patrones y su integración en el proceso de diseño, presentada en el capítulo anterior, como el modelo de representación para pdh aquí expuesto. El modelo de representación aquí expuesto ha sido utilizado para formalizar los patrones incluidos en el anexo A y que se pueden consultar en el campo “Formalización” de los mismos.

Capítulo 6

Evaluación

El objetivo de la evaluación llevada a cabo en este capítulo no es el de probar la utilidad de los patrones de diseño, y en particular la de los patrones hipermedia, cuyos méritos son avalados por su aplicación en diversos dominios y por la gran cantidad de artículos, libros y conferencias que a ellos se dedican, sino determinar tanto **la completitud como la factibilidad técnica y utilidad de la solución propuesta** a los objetivos recogidos en el capítulo 1 y los requisitos derivados de dichos objetivos definidos en el capítulo 3. Dicha especificación dio como resultado el trabajo presentado en los capítulos 4 y 5. En el primero de estos capítulos se procedió a organizar el espacio de conocimiento de los pdh dando como resultado la definición de un marco de integración de los pdh en el proceso de diseño, cuyo último fin es facilitar el paso del análisis de requisitos al diseño conceptual a través de la relación existente entre el dominio del problema (requisito) y el dominio de la solución (entidades de diseño) ofrecida por los patrones. En el segundo se procedió a definir formalmente el conocimiento subyacente a los pdh mediante un modelo de representación semántico que asienta las bases para el desarrollo de repositorios, herramientas y métodos para la organización, recuperación y exploración de los patrones de manera automatizada. La figura 6.1 muestra un esquema de las actividades de evaluación llevadas a cabo y qué partes de la solución propuesta están involucradas.

La sección 6.1 presentará la evaluación tanto del lenguaje de patrones como del marco de integración de los patrones en el proceso de diseño hipermedia (actividades V1, EF1, EA1 y EU1 en la figura 6.1). En la sección 6.2 se mostrará la evaluación del modelo de representación y el mecanismo de anotaciones semánticas definido para la formalización

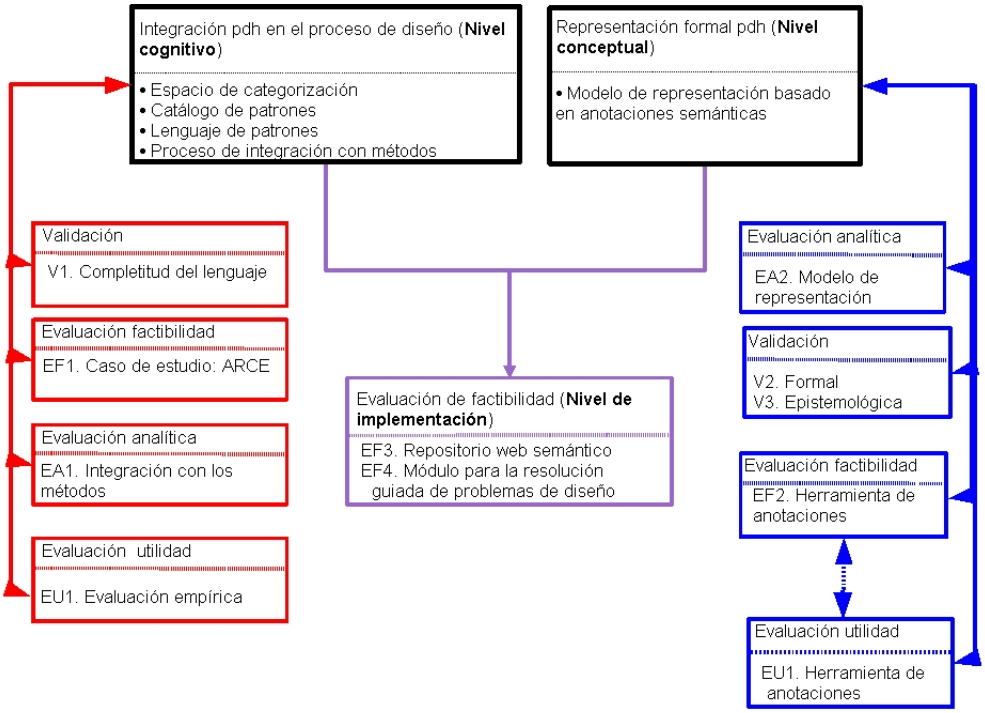


Figura 6.1: Actividades llevadas a cabo para la evaluación de esta tesis doctoral

de los patrones (actividades EA2, V2, V3, EF2 y EU1). Finalmente, en la sección 6.3 se describen dos herramientas que permiten evaluar la factibilidad técnica de la combinación del proceso de integración con la formalización de los patrones propuesta (actividades EF3 y EF4).

6.1. Evaluación del marco de integración de los pdh en el proceso de diseño

El conjunto de evaluaciones recogidas en esta sección tiene como objetivo, en primer lugar, realizar una validación de la completitud e integridad del lenguaje de patrones de diseño hipermedia propuesto en la sección 4.3 y recogido en su totalidad en el anexo A (véase la sección 6.1.1), para posteriormente poder llevar a cabo una evaluación de factibilidad del proceso de integración presentado en la sección 4.4 sobre un caso de estudio concreto (véase la sección 6.1.2) y analizar su grado de acoplamiento con respecto a los métodos de diseño hipermedia (véase la sección 6.1.3). Estas tres primeras evaluaciones permiten detectar las posibles carencias de los dos elementos que forman el marco de integración, el lenguaje y el proceso, llegando así a la evaluación de utilidad de dicho marco con un grado de madurez que permita a los usuarios detectar fallos propios del proceso y no de las herramientas que lo sustentan (véase la sección 6.1.4).

6.1.1. Validación de la completitud del lenguaje de pdh

El objetivo de este tipo de validación es determinar tanto la calidad como la naturaleza de las conexiones entre los patrones, permitiendo así conocer si el conjunto de patrones objeto de la validación es realmente un lenguaje o tan sólo una colección de patrones. Es decir, se analiza si el lenguaje aporta las relaciones necesarias para guiar al diseñador sobre cómo combinar un conjunto de patrones para resolver un problema de diseño complejo, en este caso el diseño de una aplicación hipermedia.

Para llevar a cabo esta validación se ha utilizado el test recogido en [Todd *et al.*, 2004], pero adaptado para el dominio hipermedia. El test final consta de las siguientes seis cuestiones:

1. **¿Los enlaces de referencia y contextuales entre los patrones forman un mapa?** Los enlaces de referencia son aquellas conexiones que enlazan un patrón con aquellos de menor nivel que le complementan. En el caso del lenguaje descrito en la sección 4.3 serían las relaciones referentes a "composición" y "especialización" entre patrones. El contexto de un patrón estaría formado por aquellos patrones que lo referencian, es decir, el contexto es la función inversa de las referencias. Un mapa es un gráfico creado a partir de las conexiones que aparecen en cada patrón del lenguaje.

2. **¿El mapa de contexto coincide con el mapa de referencia?** En algunos formatos de patrón los enlaces contextuales aparecen en una sección aparte, como es el caso del formato de Alexander, en la que se mencionan aquellas conexiones con patrones de mayor nivel que definen la situación funcional en la que puede ser utilizado.

3. **¿Puede el mapa ordenarse en una jerarquía de niveles?** Debería ser posible organizar los nodos dentro del mapa en una jerarquía en la que los patrones de mayor nivel proporcionasen una descripción conceptual de una aplicación hipermedia, a la vez que definen el contexto en el cual los patrones de menor nivel pueden ser utilizados.

4. **¿Pueden los niveles ser utilizados para describir una aplicación hipermedia en diferentes niveles de abstracción?** Cada uno de los niveles de la jerarquía debería poder describir una aplicación hipermedia, es decir, tratar con problemas de cada una de las vistas de diseño descritas en 2.2.1 (navegación, estructura, presentación, interacción, personalización y seguridad) desde diferentes niveles de abstracción que permitiesen abordar el diseño de estas aplicaciones de una manera iterativa.

5. **¿Cómo de "ricos" son los enlaces dentro de cada nivel de la jerarquía?** Los enlaces entre patrones del mismo nivel pueden indicar la riqueza y madurez del lenguaje. En el caso del lenguaje descrito en la sección 4.3 serían las relaciones referentes a "asociación" entre patrones. Para determinar este grado de riqueza y de madurez se definió una escala de seis niveles:

   a) Ninguna - no se observan enlaces intra-nivel en ninguno de los niveles de la jerarquía.

   b) Desconocido - muchos de los enlaces son confusos.

   c) Mínima - menos del diez por ciento de los enlaces son enlaces intra-nivel.

d) Desarrollada en un nivel - al menos una tercera parte de los enlaces de referencia dentro de un nivel son intra-nivel.

e) Desarrollada - más del diez por ciento de los enlaces son intra-nivel y ocurren en más de la mitad de los niveles debajo del raíz

f) Rica - más del treinta por ciento de los enlaces son intra-nivel y ocurre en más de la mitad de los niveles debajo del raíz.

6. **¿Pueden los patrones ser organizados por diferentes sistemas de clasificación proporcionando puntos de vista alternativos?** La existencia de varios nodos raíz en la jerarquía permite tener vistas alternativas del lenguaje.

Las cuatro primeras cuestiones permiten determinar el grado de integridad de las relaciones existentes entre el conjunto de patrones seleccionados y obtener así el estatus de lenguaje para una colección de patrones, mientras que las dos últimas determinan la completitud y madurez del lenguaje. Este test fue utilizado para evaluar el grado de madurez de tres colecciones de patrones, dos se encuentran en [33] y la tercera en [Borchers, 2000] y sólo esta última llegó a pasar el test 3 pudiendo ser clasificado como un lenguaje de patrones.

A continuación se aplicará este test al conjunto de patrones recogidos en el anexo A. En este caso, los patrones allí descritos tienen campos distintos para mencionar los patrones que conforman el contexto (véase el campo del patrón etiquetada como “Contexto”) y los patrones que referencia (véase el campo del patrón etiquetado como “Relacionado con”), por lo tanto es necesario pasar directamente al **test 2**.

Para crear el mapa es necesario identificar tres tipos de enlaces:

1. Los enlaces solamente mencionados en la sección “Contexto” aparecerán con línea punteada.
2. Los enlaces de “composición” y “especialización” solamente mencionados en la sección “Relacionado con” aparecerán con línea discontinua.
3. Los enlaces identificados en ambas secciones aparecerán con línea continua.

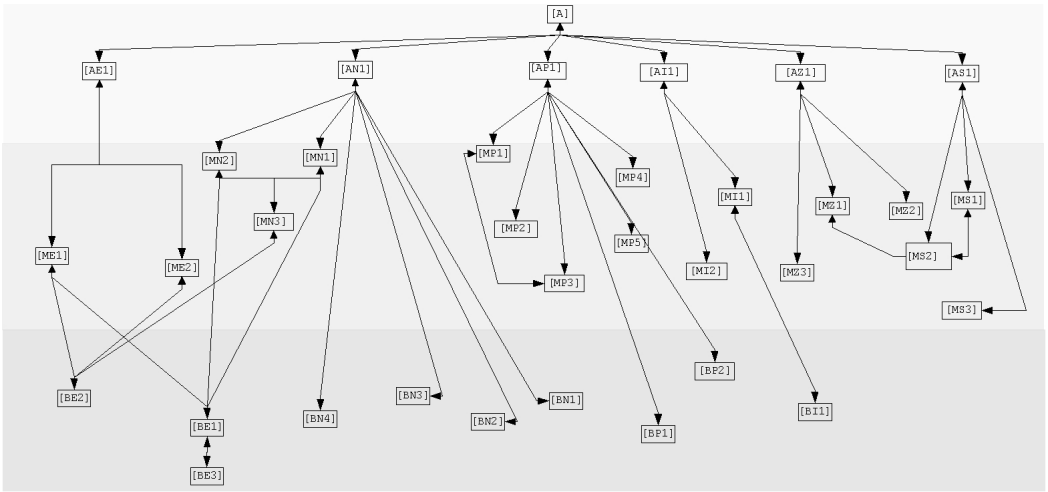
Para pasar este test, todos los enlaces deberían aparecer con líneas continuas si el mapa del contexto y el mapa de referencias coinciden. En la figura 6.2(a) se muestra el mapa creado a partir de los diferentes tipos de enlaces descritos anteriormente. Como se puede

percibir en el mapa sólo aparecen líneas continuas, por lo que se puede afirmar que esta colección de patrones **pasa el test 2**.

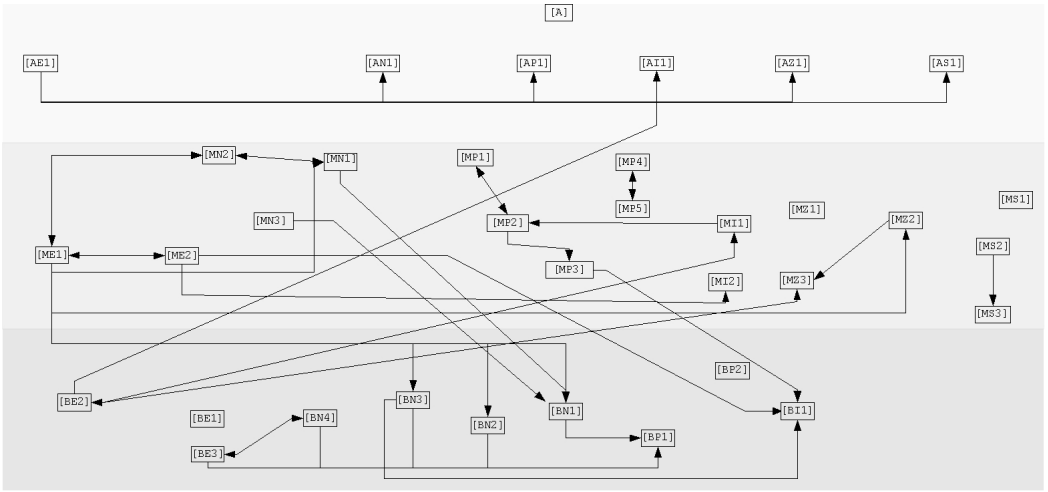
Examinando el mapa se puede apreciar que hay un patrón `[A]Hypermedia Application`, que puede actuar como raíz de una estructura jerárquica, ya que es el único que no tiene enlaces contextuales y además, se pueden reconocer tres niveles sombreados en la figura 6.2(a). El nodo raíz define el área de aplicación, el diseño de aplicaciones hipermedia. El primer nivel define las principales vistas de diseño en las cuales se puede dividir una aplicación hipermedia. El segundo nivel indica el conjunto de macro-estructuras necesarias para abordar los problemas más comunes de cada uno de esos aspectos de diseño. Finalmente, el tercer nivel define los principales componentes de las macro-estructuras recogidas en el anterior nivel. Por lo tanto se puede afirmar que este conjunto de patrones puede ser organizado de manera jerárquica, **pasando así el test 3**, aunque, como ya se comentó en la sección 4.2, existe una carencia de patrones para abordar el último nivel de abstracción para las macro-estructuras de personalización y de control de acceso.

Llegados a este punto se puede afirmar que el conjunto de patrones presentados en el anexo A tiene el **estatus de lenguaje de patrones** según la validación definida en [Todd *et al.*, 2004], es decir, existe una consistencia entre los enlaces de referencia y los enlaces de contexto dando lugar a una conectividad de los patrones del 100 %, lo cual permite a un diseñador saber cómo utilizar y combinar los patrones del lenguaje para resolver un problema de diseño. Naturalmente, este resultado ha sido fruto de un estudio iterativo que ha permitido ir refinando las relaciones entre patrones a la vez que en cada ciclo se comprobaba la integridad existente entre los campos “Contexto” y “Relacionado con”. Además, la disposición jerárquica de los patrones en tres niveles ha reafirmado el espacio de categorización propuesto en 4.1 donde se definía el criterio **Nivel de descripción**, el cual se adapta a los diferentes niveles de abstracción de los patrones hipermedia recogidos en este lenguaje y como para cada uno de esos niveles se puede describir una aplicación hipermedia a través de un conjunto de vista de diseño, recogidas en el criterio **Aspecto de diseño**.

Siguiendo con el test, cuando se definió este lenguaje de patrones en la sección 4.3 se identificaron tres tipos de enlaces. Los enlaces de composición y especialización son los que dotan al lenguaje de su estructura jerárquica, ya que definen las relaciones que permiten combinar patrones de mayor escala con aquellos de menor escala para resolver los problemas que plantean los primeros. El tercer tipo de enlace, el de asociación, define relaciones entre patrones que sugieren nuevos problemas a tratar, permitiendo así completar



(a) Mapa con las relaciones de contexto y de referencia



(b) Mapa con las relaciones intra-nivel

Figura 6.2: Validación de las relaciones del lenguaje de patrones

el problema de diseño inicial. Es este tipo de enlaces el que dota de riqueza al lenguaje, permitiendo cambiar de contexto. En la figura 6.2(b) aparecen tan sólo los enlaces intra-nivel, de los 83 enlaces contenidos en el lenguaje, 42 corresponden con este tipo de enlaces. Por lo tanto, a partir de la escala definida en el **test 5** se puede comprobar que un 50 % de los enlaces existentes en este lenguaje de patrones son intra-nivel y están repartidos a través de los diferentes niveles, dotando a este lenguaje un nivel de riqueza del tipo **"rica"**.

Finalmente, como sólo hay un nodo raíz no hay un modo alternativo para proporcionar diferentes vistas del lenguaje, por lo que a primera vista no superaría el **test 6**. Sin embargo, si se eliminara el patrón raíz y sus enlaces de referencia, los patrones que se encuentran en el primer nivel de la jerarquía pasarían a ser todos nodos raíz, ya que el único enlace que existía en su contexto era el que los relacionaba con `[A]Hypermedia Application`, proporcionando así **seis subárboles** que representarían a cada una de las vistas de diseño de una aplicación hipermedia. Esta acción puede ser llevada a cabo sin perdida de funcionalidad para el lenguaje, ya que dicho patrón, `[A]Hypermedia Application`, fue desarrollado *ex profeso* con el objetivo de dotar al lenguaje de un punto de partida que aglutinase el alcance de dicho lenguaje.

Para finalizar, hay que mencionar que desde que un lenguaje de patrones es una estructura dinámica que va evolucionando, tanto con la inclusión de nuevos patrones para proporcionar la visión de nuevos problemas o complementar los ya existentes, como con nuevas relaciones entre los patrones que se pueden descubrir a partir del uso del lenguaje, este test debe ser realizado de manera periódica y sistemática para asegurar siempre su integridad y completitud.

6.1.2. Evaluación de factibilidad

Una vez comprobada la completitud del lenguaje de patrones, es necesario comprobar su factibilidad, es decir si el lenguaje puede ser utilizado para generar diseños. Para ello se utilizará el proceso descrito en la sección 4.4 del capítulo 4 que tomaba como base el lenguaje de patrones para guiar una aproximación al modelado conceptual a través de los requisitos de una aplicación a diseñar. Dichos requisitos serán tomados de un sistema web llamado ARCE (Aplicación en Red para Casos de Emergencia, véase [10]), diseñado para mejorar la respuesta conjunta ante situaciones de emergencia entre todos los miembros de la Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil

que involucra a 21 países. Este proyecto ha sido desarrollado bajo la coordinación de la Dirección General de Protección Civil y de Emergencias de España y el grupo de investigación DEI de la Universidad Carlos III de Madrid, y en él han participado miembros de la Asociación Iberoamericana. 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 y otros mecanismos alternativos (fax y SMS) 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 [Aedo *et al.*, 2002a].

Un subconjunto de los requisitos del sistema ARCE se enumeran a continuación, y se utilizarán como resultado de llevar a cabo el **Paso 1** del proceso de integración para que los requisitos guíen una aproximación basada en patrones:

R1. En estado de normalidad los usuarios autorizados pueden publicar noticias que podrán ser leídas por cualquiera de los usuarios del sistema, e intercambiar mensajes entre ellos.

R2. En estado de emergencia, el país afectado creará un informe inicial de la situación de emergencia y podrá realizar una solicitud detallada de sus necesidades, mientras que otros países socios podrán ofertar medios movilizables y consultar la situación de las aportaciones formuladas por otros países.

R3. Todos los datos serán manejados a través de formularios que son posteriormente procesados.

R4. Existirá un histórico con todas las situaciones de emergencias cerradas.

R5. Tanto las operaciones cotidianas como las de emergencia deberán estar siempre accesibles.

R6. 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.

R7. Todas las tareas, tanto las de normalidad como las de emergencia, deberán ser activadas por usuarios autorizados. Se establecen los siguientes tipos de usuarios: **Nacionales**: Organismos asociados (Direcciones Generales de Defensa y Protección Civil), Organismos no asociados (Organismos y organizaciones invitadas); **Supranacionales**: Asociación Iberoamericana de Organismos Gubernamentales de Defensa y Protección Civil y Organismos Internacionales.

R8. 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 la interfaz de usuario como por el glosario de términos.

El **Paso 2** lleva a asignar a cada requisito aquel patrón que mejor lo cubre. Esta asociación de un problema específico de diseño, que describe el requisito, a un problema genérico de diseño, que describe el patrón, va a permitir que sea el patrón el que guíe el paso hacia una representación conceptual del requisito facilitando su posterior implementación en el sistema final, y que en otras circustancias, esta transformación dependería de la experiencia y conocimientos del diseñador. En la tabla 6.1 puede verse el resultado de realizar dicha tarea sobre los requisitos anteriormente descritos.

R1	[ME2]Task-based Organization Aspecto de diseño: Estructural Nivel de descripción: Medio
R2	[ME2]Task-based Organization Aspecto de diseño: Estructural Nivel de descripción: Medio
R3	[MP1]Interaction -Information Decoupling Aspecto de diseño: Presentación Nivel de descripción: Medio
R4	[BE2]Node as a Single Unit Aspecto de diseño: Estructural Nivel de descripción: Bajo
R5	[MN1]Index Navigation Aspecto de diseño: Navegación Nivel de descripción: Medio
R6	[BE2]Node as a Single Unit Aspecto de diseño: Estructural Nivel de descripción: Medio

R7	[MS2]Role-Based Access Control Aspecto de diseño: Seguridad Nivel de descripción: Medio
R8	[MZ3]Content Personalization Aspecto de diseño: Personalización Nivel de descripción: Medio

Tabla 6.1: Patrones seleccionados después de aplicar el paso 2 a los requisitos del sistema ARCE

Como todos los patrones seleccionados recaen en un nivel de descripción medio o bajo, se puede llevar a cabo el siguiente paso.

El **Paso 3** lleva a refinar los patrones ya seleccionados con el objetivo de completar las soluciones que éstos proponen, siguiendo las relaciones de composición y generalización, o sugerir nuevos problemas, siguiendo las relaciones de asociación, que podrían provocar la incorporación de un nuevo requisito para el sistema. Todas estas relaciones están recogidas en el mapa del lenguaje de la figura 4.4. Este paso se realiza de manera iterativa hasta que todos los patrones seleccionados recaigan en los niveles de descripción medio y bajo. En la tabla 6.2 se han incorporado a los patrones ya recogidos en la tabla 6.1 los seleccionados tras realizar esta tarea.

R1	[ME2]Task-based Organization Aspecto de diseño: Estructural Nivel de descripción: Medio [BE2]Node as a Single Unit Aspecto de diseño: Estructural Nivel de descripción: Bajo
R2	[ME2]Task-based Organization Aspecto de diseño: Estructural Nivel de descripción: Medio [BE2]Node as a Single Unit Nivel de descripción: Bajo

R3	[MP1]Information – Interaction Decoupling Aspecto de diseño: Presentación Nivel de descripción: Medio
R4	[BE2]Node as a Single Unit Aspecto de diseño: Estructural Nivel de descripción: Bajo [MI1]Information on Demand Aspecto de diseño: Interacción Nivel de descripción: Medio [BI1]Action Button Aspecto de diseño: Interacción Nivel de descripción: Bajo
R5	[MN1]Index Navigation Aspecto de diseño: Navegación Nivel de descripción: Medio [BE3]Home Page Aspecto de diseño: Estructural Nivel de descripción: Bajo [BP1]Navigation Bar Aspecto de diseño: Presentación Nivel de descripción: Bajo
R6	[BE2]Node as a Single Unit Aspecto de diseño: Estructural Nivel de descripción: Bajo
R7	[MS2]Role-Based Access Control Aspecto de diseño: Seguridad Nivel de descripción: Medio [MZ1]Role Subtype Aspecto de diseño: Personalización Nivel de descripción: Medio

R8	[MZ3]Content Personalization Aspecto de diseño: Personalización Nivel de descripción: Medio

Tabla 6.2: Patrones seleccionados después de aplicar el paso 3 a los requisitos del sistema ARCE

Antes de pasar al siguiente paso, hay que destacar dos decisiones que se tomaron. Primero, en el requisito R4, se ha planteado la incorporación del patrón [MI1]Information on Demand que incorpora la necesidad de que el usuario pudiese controlar la cantidad de información que desea recibir en la pantalla, y así pueda decidir con qué nivel de detalle quiere visualizar el histórico de emergencias. La segunda decisión fue en el requisito R5, decidiéndose que era mejor elegir la especialización del patrón [BE1]Collection Center, el patrón [BE3]Home Page, y tener así acceso a todas las tareas del sistema desde la página principal.

El **Paso 4** tiene como objeto organizar los requisitos de acuerdo con el aspecto de diseño al que pertenece el patrón (estructura, navegación, presentación, interacción, personalización y seguridad) y los patrones de cada aspecto según las dependencias que existan entre ellos, es decir qué patrón necesita de otro para llevar a cabo la solución que propone.

Estructura	
R1	[ME2]Task-based Organization [BE2]Node as a Single Unit
R2	[ME2]Task-based Organization [BE2]Node as a Single Unit
R4	[BE2]Node as a Single Unit
R5	[BE3]Home Page
R6	[BE2]Node as a Single Unit
Navegación	
R5	[MN1]Index Navigation
Presentación	
R3	[MP1]Information-Interaction Decoupling
R5	[BP1]Navigation Bar
Interacción	

R4	[MI1]Information on Demand [BI1]Action Button
Personalización	
R7	[MZ1]Role Subtype
R8	[MZ3]Content Personalization
Seguridad	
R7	[MS2]Role-Based Access Control

Tabla 6.3: Patrones organizados después de aplicar el paso 4 a los requisitos del sistema ARCE

El **Pasos 5** tienen como objeto adaptar la solución de los patrones al dominio de la aplicación, identificando aquellos elementos de diseño que participan en la construcción de un esquema de solución independiente de métodos de diseño. Como no existe una representación de referencia común a todos ellos, se utilizará la ontología del dominio hipermedia descrita en la sección 5.2.2 como representación intermedia para su transformación final en el Paso 6 a entidades de diseño propias del método que esté utilizando el diseñador. A continuación, se irá mostrando el esquema resultante para cada uno de los aspectos de diseño a partir de la formalización asociada a cada uno de los patrones seleccionados, y que se recogen en el campo "Formalización" del anexo A.

Estructura

Del paso 4 se obtiene que los requisitos R1, R2, R4, R5 y R6 recogen las necesidades del usuario en cuanto a las tareas que espera poder realizar y la información a consultar en el sistema. Como soluciones parejas a los requisitos R1 y R2, se propone una combinación de dos patrones `[ME2]Task-based Organization` y `[BE2]Node as a Single Unit` que permite organizar las tareas de la aplicación. Para el requisito R1 se generaría el siguiente esquema, un nodo compuesto que contiene cada una de las tareas relacionadas para el estado de normalidad:

```
CompositeComponent(Normalidad)
relationship("aggregation")
hasComponents(Normalidad,PublicarNoticias)
hasComponents(Normalidad,IntercambiarMensajes)
Node(PublicarNoticias)
Node(IntercambiarNoticias)
```

De igual manera sucede para el requisito R2, que engloba las tareas en estado de emergencia:

```
CompositeComponent(Emergencia)
relationship("aggregation")
hasComponents(Emergencia,InformeInicial)
hasComponents(Emergencia,SolicitudDetallada)
hasComponents(Emergencia,ListadoAportaciones)
Node(InformeInicial)
Node(SolicitudDetallada)
Node(ListadoAportaciones)
```

El requisito R4 y R6 son resueltos de la misma manera, en su aspecto estructural, por el patrón `[BE2]Node as a Single Unit`, que describe a un nodo que visualizará el histórico de emergencias, y el glosario respectivamente:

```
Node(Historico)
Node(Glosario)
```

Finalmente, el requisito R5, en su parte estructural, es resuelto por el patrón `[BE3]Home Page`, el cual describe un nodo que servirá de acceso a toda la aplicación, al cual llamaremos Arce, y está formado por un esquema de navegación y un espacio para mostrar la información que, por lo tanto, agrupa a todos los nodos definidos anteriormente:

```
Root(Arce)
relationship("aggregation")
hasComponents(Arce,Informacion)
hasComponents(Arce,EsquemaNavegacion)
CompositeComponent(Informacion)
Node(EsquemaNavegacion)
relationship("generalization")
hasComponents(Informacion,InformeInicial)
hasComponents(Informacion,SolicitudDetallada)
hasComponents(Informacion,ListadoAportaciones)
hasComponents(Informacion,PublicarNoticias)
hasComponents(Informacion,IntercambiarNoticias)
```

Navegación

En este aspecto, tan sólo se encuentra el requisito R5 que recogía la necesidad de tener siempre accesibles las tareas del sistema, el cual es resuelto por el patrón `[MN1]Index Navigation`. Este patrón propone definir enlaces desde un punto de entrada a cada uno de los nodos que quieren ser accedidos. Recordemos que ese "punto de entrada" ya ha sido definido en el aspecto de diseño anterior, dentro del nodo raíz Arce, como el nodo "EsquemaNavegacion". El esquema resultante sería:

```
Link(ArceToNormalidad)
hasSource(ArceToNormalidad,AnclaArceNormalidad)
hasTarget(ArceToNormalidad,AnclaNormalida)
Anchor(AnclaArceNormalidad)
hasAnchors(EsquemaNavegacion,AnclaArceNormalidad)
Anchor(AnclaNormalidad)
hasAnchors(Normalidad,AnclaNormalidad)
Link(ArceToEmergencia)
hasSource(ArceToEmergencia,AnclaArceEmergencia)
hasTarget(ArceToEmergencia,AnclaEmergencia)
Anchor(AnclaArceEmergencia)
hasAnchors(EsquemaNavegacion,AnclaArceEmergencia)
Anchor(AnclaEmergencia)
hasAnchors(Emergencia,AnclaEmergencia)
Link(ArceToHistorico)
hasSource(ArceToHistorico,AnclaArceHistorico)
hasTarget(ArceToHistorico,AnclaHistorico)
Anchor(AnclaArceHistorico)
hasAnchors(EsquemaNavegacion,AnclaArceHistorico)
Anchor(AnclaHistorico)
hasAnchors(Historico,AnclaHistorico)
Link(ArceToGlosario)
hasSource(ArceToGlosario,AnclaArceGlosario)
hasTarget(ArceToGlosario,AnclaGlosario)
Anchor(AnclaArceGlosario)
hasAnchors(EsquemaNavegacion,AnclaArceGlosario)
Anchor(AnclaGlosario)
hasAnchors(Glosario,AnclaGlosario)
```

Presentación

El requisito R3 recogía la necesidad de que todos lo datos fuesen recogidos mediante formularios, y para ayudar al usuario a diferenciar entre información e interacción el patrón [MP1]Information-Interaction Decoupling define diferentes areas de contenido dentro de un nodo según funcionalidades. Por lo tanto, para los nodos que contienen este tipo de formularios se definen los siguientes tipos de contenidos:

```
hasContents(Informacion,Etiquetas)
hasContents(Informacion,CamposFormulario)
hasContents(Informacion,Botones)
Content(Etiquetas)
Content(CamposFormulario)
Content(Botones)
```

La última parte para la solución del requisito R5 se resuelve con el patrón [BP1]Navigation Bar, como parte del patrón [BE3]Home Page que se aplicó en el aspecto estructural:

```
Content(ContenidoNormalidad)
hasContents(EsquemaNavegacion,ContenidoNormalidad)
hasLocation(AnclaArceNormalidad,ContenidoNormalidad)
Content(ContenidoEmergencia)
hasContents(EsquemaNavegacion,ContenidoEmergencia)
hasLocation(AnclaArceEmergencia, ContenidoEmergencia)
Content(ContenidoHistorico)
hasContents(EsquemaNavegacion,ContenidoHistorico)
hasLocation(AnclaArceHistorico,ContenidoHistorico)
Content(ContenidoGlosario)
hasContents(EsquemaNavegacion,ContenidoGlosario)
hasLocation(AnclaArceGlosario,ContenidoGlosario)
```

Interacción

El patrón [MI1]Information on Demand en su solución expone presentar un subconjunto de los contenidos de un nodo y dejar al usuario que controle qué información adicional necesita mediante un botón de activación. Este botón se puede representar con la ayuda del patrón [BI1]Action Button. Las emergencias cerradas que recoge el requisito R4 podrán ser visualizadas según el usuario las demande:

```
Node(Historico)
```

```
hasContents(Historico,IDEmergencia)
hasContents(Historico,InformacionGeneral)
hasContents(Historico,Boton)
Content(Boton)
hasEvents(Boton,MasInformacion)
Event(MasInformacion)
```

Personalización

El patrón `[MZ1]Role Subtype` en su solución identifica a cada tipo de usuario de la aplicación con un rol y define una jerarquía en la que los roles generales son especializados en roles más concretos. Por lo tanto, a partir de los tipos de usuarios especificados en el requisito R7 se obtiene la siguiente instanciación del patrón:

```
User(UsuariosAutorizados)
userType(UsuariosAutorizados,"role")
userList(UsuariosAutorizados,Nacionales)
userList(UsuariosAutorizados,SupraNacionales)
User(Nacionales)
userType(Nacionales,"role")
userList(Nacionales,OrganismosAsociados)
userList(Nacionales,OrganismosNoAsociados)
User(OrganismosAsociados)
userType(OrganismosAsociados,"role")
userList(OrganismosAsociados,DireccionGeneralDefensa)
userList(OrganismosAsociados,ProteccionCivil)
User(DireccionGeneralDefensa)
User(ProteccionCivil)
User(OrganismosNoAsociados)
userType(OrganismosNoAsociados,"role")
userList(OrganismosNoAsociados,InstitucionesPublicas)
userList(OrganismosNoAsociados,ONG)
User(InstitucionesPublicas)
User(ONG)
User(SupraNacionales)
userType(SupraNacionales,"role")
userList(SupraNacionales,AIOGDPC)
userList(SupraNacionales,OrganismosInternacionales)
```

```
User(AIOGDPC)
User(OrganismosInternacionales)
```

Con respecto al requisito R8, el patrón `[MZ3]Content Personalization` propone que el valor de los contenidos de los nodos varíe según el rol del usuario.

```
Attribute(A1)
name(A1,idioma)
hasAttributes(Arce,A1)
hasAttributes(UsuariosAutorizados,A1)
Event(ContenidoIdioma)
hasEvents(Arce,ContenidoIdioma)
```

Seguridad

El patrón `[MS2]Role-Based Access Control` propone que a cada rol se le indique si puede acceder a una determinada información. El requisito R7 especifica que todas las tareas del sistema sólo serán accedidas por los usuarios autorizados, por lo que se obtiene el siguiente esquema:

```
accessList(UsuariosAutorizados,Arce)
```

Para finalizar esta sección, es preciso 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.

El último paso de este proceso tiene como objetivo instanciar los esquemas de solución aquí propuestos a entidades concretas de diseño. Este paso se verá en la sección 6.3.2, en el cual se ha implementado el proceso aquí llevado a cabo en una herramienta de soporte para un método de diseño concreto.

6.1.3. Evaluación analítica

En la sección anterior se ha desarrollado cómo se aplicaría el proceso de integración de los patrones en el proceso de diseño basado en los requisitos de un caso determinado. Los cinco primeros pasos son independientes del método de diseño en el que se vaya a describir las soluciones propuestas por los patrones y es en el Paso 6 en el que se produce la transformación del esquema propuesto a entidades de diseño propias del método, con la entrada ordenada a cada uno de sus productos. Esta sección tiene como objeto analizar la capacidad de los métodos de diseño para recibir los requisitos categorizados según el aspecto de diseño donde tienen implicaciones, es decir si cuentan con productos que cubran cada una de los esquemas de solución propuestos en el Paso 5.

La tabla 6.4 recoge en cada una de sus filas los métodos de diseño descritos en la sección 2.2.3 y los productos con los que cuentan para plasmar las soluciones propuestas.

A partir de lo analizado en la tabla 6.4, se puede observar que ADM sería el método que podría recibir una entrada ordenada de los requisitos y plasmar así las soluciones propuestas por los patrones para dichos requisitos de manera completa. Del resto de métodos, un 60 % podría describir los esquemas de solución propuestos con sus productos, excepto en el caso de los problemas de diseño relacionados con la seguridad de la aplicación. Esto evidencia, por un lado, que el lenguaje de patrones aquí recogido no puede ser utilizado en su totalidad por la mayoría de los métodos, a la vez que se pone de manifiesto que dichos métodos no cuentan con productos que permitan plasmar los problemas relacionados con este aspecto de diseño, tan relevante en los sistemas multi-usuario, como son los sistemas web.

6.1.4. Evaluación de utilidad

Para completar el conjunto de evaluaciones presentadas en las anteriores secciones, en las cuales se ha medido la completitud y la integridad del lenguaje de patrones para guiar al diseñador en cómo combinar los patrones para resolver el diseño de una aplicación hipermedia y la factibilidad del proceso de integración propuesto para generar diseños conceptuales dentro del proceso de diseño llevado a cabo por métodos, se decidió también realizar una evaluación de utilidad sobre estas cuestiones con el objetivo de medir el grado de satisfacción de un conjunto de evaluadores. Los evaluadores fueron estudiantes de 5º curso de la titulación Ingeniería Informática de la Universidad Carlos III de Madrid, y el proceso

Métodos	Navegación	Presentación	Estructura	Comportamiento	Personalización	Seguridad
RMM	Diagrama de Slice	Diagrama de Slice	Diagrama E-R	-	-	-
OOHDM	Modelo de Navegación	Modelo de Interfaz Abstracto	Modelo Conceptual	Modelo de Interfaz Abstracto	Vistas del modelado, Perfiles usuario	-
WSDM	Modelo de Navegación Conceptual	Modelo de Implementación	Modelo de Objeto de Usuario	-	Todos sus productos	-
WebML	Modelo del Hipertexto	Modelo de Presentación	Modelo Estructural	Modelo del Hipertexto	Unidades de Operación	-
OO-H	Diagrama de Acceso Navegacional	Diagrama de Presentación Abstracto	Diagrama de Clase UML	Diagrama de Clase UML	Diagrama de Clase UML, Reglas de Personalización	-
ADM	Diagrama de Navegación	Diagrama Interno	Diagrama Estructural	Catálogo de Eventos	Diagrama de Usuarios, Catálogo de Categorización, Tabla de Acceso	Diagrama de Usuarios, Catálogo de Categorización, Tabla de Acceso

Tabla 6.4: Análisis de los métodos con respecto a los aspectos de diseño

de evaluación se desarrolló durante el transcurso de la asignatura optativa Diseño y Evaluación de Sistemas Hipermedia. Dichos evaluadores, seis parejas de diseñadores, tenían conocimientos de cómo utilizar un lenguaje de patrones, en concreto el recogido en el libro [van Duyne *et al.*, 2002] para el diseño de interfaces web, y en el diseño de aplicaciones hipermedia utilizando el método ADM [Díaz *et al.*, To be published], pero no en cómo aplicar patrones para la generación de diseños, como puedan ser los patrones de diseño de orientación a objetos recogidos en [Gamma *et al.*, 1994], por lo tanto se puede considerar que el grupo tenía una experiencia de grado medio en el dominio de patrones y en el diseño hipermedia.

Previamente a la sesión de evaluación, los alumnos recibieron un documento con el lenguaje de patrones (véase el anexo A), para que pudiesen familiarizarse con ellos, y se les presentó tanto los pasos a seguir para utilizar el lenguaje en el proceso de diseño hipermedia (véase la sección 4.4) como el caso sobre el cual iban a trabajar, un conjunto de los requisitos más generales del sistema ARCE (véase la sección 6.1.2).

Con estas herramientas debían generar un diseño conceptual del sistema ARCE basado en el método ADM y cumplimentar posteriormente el cuestionario recogido en el anexo C.1. El objetivo del cuestionario era, por un lado, percibir el grado de satisfacción del usuario hacia el lenguaje de patrones en términos de su estructura y formato y, por otro, detectar las dificultades encontradas a la hora de realizar la tarea propuesta. Dentro del cuestionario había preguntas abiertas y preguntas con una escala de cinco puntos (1=mala y 5=muy buena).

La primera pregunta **¿Qué opinión te merece el lenguaje con respecto a las siguientes características?**, trataba cuestiones generales como la utilidad, completitud, organización, estilo narrrativo, formato y representación de la solución de los patrones. Como resultado en la figura 6.3 se puede observar que el lenguaje de patrones goza de una aceptación “buena” en sus diferentes aspectos.

Posteriormente con la cuestión **¿Qué opinión te merece el lenguaje con respecto a las siguientes características?** se profundizaba en la estructura del lenguaje preguntando por la utilidad y usabilidad de cada uno de los grupos en los que se encuentra dividido el lenguaje. En la figura 6.4 puede observarse que cada grupo de patrones por separado también tiene una aceptación “buena”, aunque los grupos `[AP1]Aesthetics` y `[AI1]Interaction` son menos valorados.

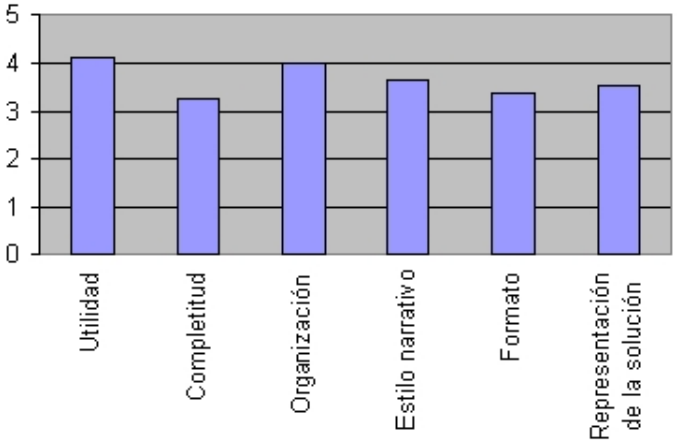


Figura 6.3: Análisis general sobre el lenguaje de patrones

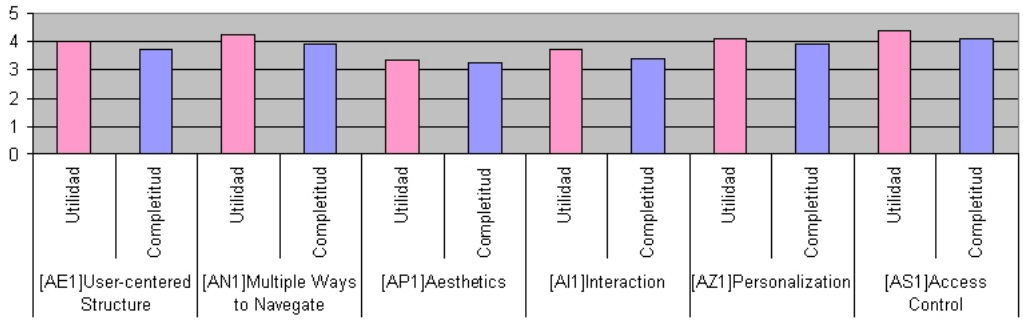


Figura 6.4: Análisis de cada una de las categorías del lenguaje de patrones

En cuanto a la cuestión abierta **¿Has encontrado algún aspecto que mejorarías? (Eliminar algún patrón, añadir, reescribir, nombre del patrón etc.)** se recogieron las siguientes sugerencias:

1. Añadir más patrones de bajo nivel como los utilizados en [van Duyne *et al.*, 2002].

2. Cambiar el formato de identificador de los patrones, pasando de [Nivel descripción Aspecto de diseño Número] a [Aspecto de diseño Nivel descripción Número].

3. Utilizar gráficos más simples para ayudar a su compresión.

La sugerencia número tres tuvo como consecuencia la elaboración de los gráficos que aparecen en el campo solución de los patrones del anexo A, mostrando todos el mismo aspecto al estilo de los que aparecen en los libros [Alexander *et al.*, 1977] y [van Duyne *et al.*, 2002].

Con respecto al proceso para llevar a cabo el paso desde los requisitos del sistema a una primera aproximación del modelado conceptual guiado por patrones, a la cuestión **¿Has tenido problemas a la hora de identifcar el patrón que cubría a cada requisito? ¿Cuáles?**, cuatro de los evaluadores respondieron que habían tenido problemas con los requisitos R3 y R4 para encontrar un patrón adecuado. Esto se puede deber a que dichos requisitos, en particular, son demasiado específicos y las intenciones de los patrones demasiado genéricas, por lo que era requerida una mayor profundización en la descripción y conocimientos de los patrones.

Ante la cuestión **¿Has tenido problemas a la hora de transformar la solución del patrón a una estructura de diseño concreta? ¿Cuáles?**, respondieron que no, e incluso mencionaron que los más sencillos de aplicar fueron los patrones relacionados con los tipos de usuario y la seguridad.

Como cuestiones finales al proceso de aplicación se realizaron las preguntas **¿Estás satisfecho con el resultado final del modelado?** y **¿Hubiéses preferido realizar el modelado de la aplicación sin el uso del lenguaje de patrones?**. En la figura 6.5 puede observarse que la gran mayoría están satisfechos con el resultado de su modelado. En la figura 6.6 se observa que las respuestas de los evaluadores están divididas entre que les hubiese dado igual utilizar el lenguaje o no para el modelado (valor “medio”) con los que sí parecen preferir su utilización (valor “no mucho”). Esto se puede deber a los comentarios que dieron algunos de los evaluadores al final del cuestionario sobre la utilización de los patrones cuando se tiene o no experiencia en el diseño de aplicaciones. Estos evaluadores opinaban que los patrones eran de gran ayuda cuando no se tenía experiencia, pero que cuando uno la adquiría daba más importancia a su propia experiencia que a lo que exponían los patrones “no es necesario seguir de una manera estricta lo que dicen los patrones”. Paradógicamente, es la experiencia de los diseñadores la que hace de los patrones algo dinámico y en continuo cambio, ya que como Alexander expuso, el último paso en la selección de un patrón es que el propio diseñador añada su propio material, modificando los patrones existentes para que sean más relevantes a la situación actual o creando nuevos patrones [Alexander *et al.*, 1977].

Finalmente, cerrando el cuestionario, la pregunta **¿Has utilizado el nombre de los patrones para comunicarte con el resto de los miembros del equipo, por ejemplo para**

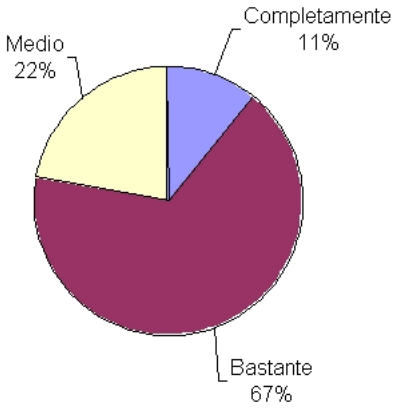


Figura 6.5: Análisis sobre la satisfacción personal del resultado obtenido

discutir cuestiones sobre el diseño? tenía como objeto conocer si una de las cualidades más importantes de los patrones, su nombre, había afectado en el modo de comunicación de los evaluadores durante el proceso. En la figura 6.7 se puede observar que los evaluadores vuelven a estar divididos entre el valor “bastante” y “medio”.

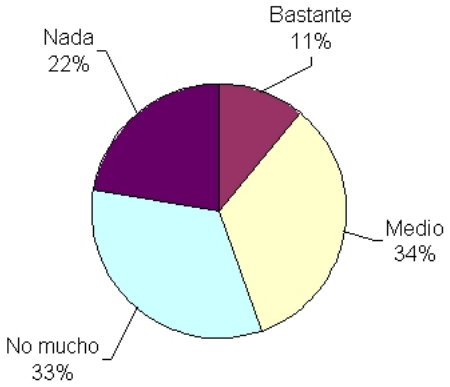


Figura 6.6: Análisis de la no utilización del proceso

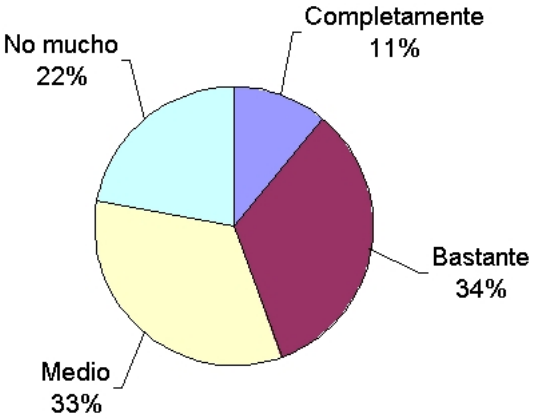


Figura 6.7: Análisis de la utilización del nombre del patrón como medio de comunicación

6.2. Evaluación del modelo de representación formal

Esta sección recoge el conjunto de evaluaciones que tienen como objeto tanto la validación del modelo de representación descrito en el capítulo 5, como su utilidad para la formalización de patrones por usuarios sin conocimientos del dominio ontológico.

En primer lugar, se ha llevado a cabo una evaluación analítica que comprueba la idoneidad del mecanismo de representación formal elegido, las ontologías, y del modelo de representación, con respecto a una serie de criterios que deberían ser tenidos en cuenta a la hora de acometer la representación formal de los patrones de diseño hipermedia, además de comparar dicha representación con otras propuestas de formalización de patrones de diseño, que fueron recogidas en la sección 2.1.7 "Formalización de patrones", utilizando para ello los mismos criterios (véase la sección 6.2.1).

En segundo lugar, puesto que uno de los objetivos de la representación formal es la creación de repositorios de conocimiento de patrones de diseño hipermedia a partir de los cuales se puedan desarrollar aplicaciones software, se ha procedido a realizar primeramente, la validación del modelo de conocimiento con respecto a su definición subyacente en lógica de descripción, y comprobar si es satisfacible. A continuación, se ha procedido a formalizar cada uno de los patrones recogidos en el anexo A para comprobar la suficiencia de la expresividad del modelo de representación (véase la sección 6.2.2).

En tercer lugar, se ha llevado a cabo una evaluación para conocer el grado de utilidad y de satisfacción de diseñadores y usuarios sobre la formalización de los patrones de diseño hipermedia mediante anotaciones semánticas, el cual es soportado tecnológicamente por AnnotPat (véase la sección 6.2.3).

6.2.1. Evaluación analítica

En la sección 3.3 "Requisitos para una integración de patrones en el proceso de diseño hipermedia" se enumeraron una serie de requisitos que debería tener en cuenta un modelo de representación para patrones de diseño hipermedia debido, por un lado, a las **características especiales del dominio a representar**, como es la heterogeneidad en la descripción del patrón que hace que algunos patrones no puedan ser aplicados de manera automática en herramientas de diseño por poseer un nivel de abstracción alto, cercano a la perspectiva que tiene el usuario final de la aplicación, y la no existencia de un modelo de referencia

Requisitos	UML	Lógica matemática	Modelo ontológico
R5.1 Formato del patrón	Solución	Solución	Todo el formato
R5.2 Relaciones entre patrones	No	No	Si
R5.3 Elementos del dominio	Modelado OO	Modelado OO	Modelado hipermedia
R5.4 Extensible formato y dominio	No	No	Si
R5.5 Abstracto	Orientado a codificación	Orientado a codificación	Orientado a codificación y exploración
R5.6 Repositorios	No	No	Si
R5.7 Interoperabilidad	No	No	Si
R5.8 Entendible por usuarios	Conocimientos OO y UML	Conocimientos matemáticos	Conocimientos modelado hipermedia

Tabla 6.5: Análisis de las aproximaciones de formalización de patrones

común para representar las soluciones de los patrones; y por otro, a **la intención de dicha representación formal**, como es el desarrollo de herramientas tanto para la aplicación de patrones como su exploración. Estos requisitos se han utilizado para comparar el modelo de representación descrito en el capítulo 5 con el resto de propuestas para la formalización de patrones de diseño, recogidas en la sección 2.1.7 “Formalización de patrones” y que fueron agrupadas según el mecanismo de representación formal utilizado para su definición (UML o lógica matemática).

El modelo de representación propuesto cumple con los requisitos exigibles a una representación formal para la representación de patrones de diseño hipermedia, como queda recogido en la tabla 6.5:

- **R5.1 Formato del patrón**. El modelo de representación propuesto tiene como subcomponente la descripción de cada uno de los elementos de la estructura del patrón. El resto de propuestas tan sólo tienen en cuenta la representación formal de un elemento, la solución del patrón.
- **R5.2 Relaciones entre patrones**. El modelo de representación propuesto tiene en cuenta las relaciones existente entre patrones, tanto a nivel de contexto como a nivel de patrones relacionados, lo que permite representar un lenguajes de patrones. Como

consecuencia de no cumplir el requisito anterior, el resto de propuestas sólo permiten representar el patrón como elemento independiente.

- **R5.3 Elementos del dominio**. Naturalmente, el modelo aquí propuesto recoge los elementos del dominio hipermedia que permiten representar el problema y la solución de los patrones de diseño hipermedia, al igual que las otras aproximaciones representan los elementos necesarios del dominio de la orientación a objetos para representar la solución por el patrón. Sin embargo, la arquitectura propuesta separa la definición de los elementos del dominio de la definición del formato del patrón, lo que posibilita utilizar este modelo para el dominio de la orientación a objetos mediante la definición de una ontología que conceptualice dicho conocimiento.

- **R5.4 Extensible formato y dominio**. La arquitectura del modelo se ha definido a dos niveles. En el primer nivel se definen de manera independiente tanto el formato del patrón como los elementos para el dominio de las aplicaciones hipermedia. Posteriormente, en un segundo nivel se combinan ambos componentes para definir el concepto de patrón de diseño hipermedia. Esta representación permite la incorporación de nuevos elementos tanto en el formato como en el dominio de aplicación sin que afecte a la estructura final de patrón de diseño hipermedia. Además posibilita la reutilización del modelo de representación por patrones de otros dominios. Con respecto al resto de aproximaciones la extensión de su aproximación no está contemplada al no tener como elementos variables el formato del patrón y los elementos del dominio, cosa que sí sucede en el área hipermedia.

- **R5.5 Abstracto**. Con abstracto se indica que la representación formal debería ser independiente del objeto de su aplicación, es decir, si va a ser utilizada para aplicar un patrón en una herramienta de diseño [France *et al.*, 2004], o para verificar su existencia en un determinado lenguaje de programación [Eden *et al.*, 1997], y ser representar el concepto en sí. El modelo aquí presentado ha seguido una aproximación basada en la representación del conocimiento, en la que los elementos que conforman el patrón han sido investigados de manera independiente para su conceptualización. Posteriormente, como se verá en la sección 6.3 se procederá a comprobar su factibilidad técnica mediante el desarrollo de dos herramientas de objetivos diferentes, como pueda ser la exploración de patrones y su aplicación en una herramienta de diseño.

- **R5.6 Repositorios**. Junto al modelo se ha desarrollado una herramienta que permite la formalización de patrones de diseño según el modelo propuesto. Estos patrones

son gestionados mediante un repositorio que puede ser utilizado por programas software. El resto de aproximaciones desarrollan herramientas *ad-hoc* que prueban la factibilidad técnica de dichos modelos pero no el desarrollo de repositorios para su compartición.

- **R5.7 Interoperabilidad**. Este requisito es fundamental para poder integrar la formalización de los patrones de diseño hipermedia en los métodos, ya que como se ha mencionado los métodos no comparten el mismo modelo de referencia y por lo tanto la solución de cada patrón de diseño tiene una representación diferente en cada uno de ellos. Una de las principales cualidades de las ontologías es la interoperabilidad entre representaciones que pueden ser compartidas y reutilizadas, permitiendo que una determinada conceptualización sea interpretada con un significado propio en diferentes contextos. El modelo de representación para los elementos de diseño hipermedia aquí propuesto ofrece un marco a partir del cual cada uno de los métodos puede describir sus entidades de diseño en función de éste y mantener así su independencia, como se verá en la sección 6.3.2 en la que se muestra la integración de los patrones de diseño hipermedia en una herramienta de soporte de un método concreto. Las otras aproximaciones no fueron definidas con este objetivo, ya que en el área de la orientación a objetos se comparte una única semántica para el modelado conceptual de estas aplicaciones.

- **R5.8 Entendible por usuarios**. Las formalizaciones también deberían ser herramientas que los propios destinatarios de su aplicación pudiesen utilizar para definir sus propios repositorios de datos. Las anotaciones semánticas se han utilizado en esta propuesta con el objeto de hacer transparente la formalización de los patrones de diseño hipermedia tanto a diseñadores como a autores. Para ello se ha desarrollado una herramienta, AnnotPat (véase la sección 5.3.3), que permite al usuario seleccionar partes de la descripción del patrón y asignarles su correspondiente semántica, a partir de la cual se pueden generar patrones de diseño hipermedia entendibles por programas software. En las aproximaciones basadas en UML, el usuario debe tener conocimientos de ingeniería del software y del propio lenguaje. Mientras que esta circunstancia puede parecer trivial dentro del área software, en la ingeniería hipermedia no todos los diseñadores son ingenieros software, ya que podemos encontrar a diseñadores gráficos o a los autores de los contenidos, y menos aún poseerán conocimientos en lógica matemática.

Como conclusión a esta comparativa se puede destacar que el hecho de que las aproximaciones basadas en UML o lógica matemática se encuentren en dominios de aplicación diferentes no debería ser motivo para no poder ser reutilizadas. Sin embargo, debido principalmente a la falta de cumplimiento en los requisitos R5.1 y R5.5 de las aproximaciones de la ingeniería del software, se planteó una definición que tuviese en cuenta el formato completo del patrón para el desarrollar aplicaciones para la exploración y búsqueda de patrones, salvaguardando así la descripción textual del patrón, eje de la cualidad comunicativa de los mismos.

6.2.2. Validación formal y epistemológica del modelo

A la hora de validar un modelo formal de conocimiento es necesario evaluar tanto su aspecto formal como su capacidad expresiva para formalizar dicho conocimiento.

En el primer caso, la validez formal del modelo de representación necesita comprobar que el conocimiento en él reflejado no presenta contradicciones, es decir está bien especificado utilizado el lenguaje formal escogido. En el caso de esta tesis, se eligió OWL, por ser el estándar para la web semántica, además de poder aprovechar todos los esfuerzos que se están realizando en el desarrollo de herramientas para su gestión, los cuales se han utilizado como tecnología base para el posterior desarrollo de las diferentes herramientas contenidas en esta tesis. Dentro de los lenguajes que conforman OWL, se eligió OWL-DL (*Description Logic*) ya que al estar basado en lógica de descripción puede ser procesado por un razonador (véase el anexo B para la descripción de OWL-DL y su equivalencia entre expresiones en lógica de descripción y los elementos del lenguaje).

Para ayudar al desarrollo de las ontologías y su posterior validación formal se utilizó el editor Protégé con el *plugin* OWL en su versión 2.0 [27] y el razonador de lógica de descripción RACER en su versión 1.7.24 [18]. En el editor fueron implementadas las ontologías “Componentes de un patrón”, del “Dominio hipermedia” y “Patrón de diseño hipermedia”. Posteriormente, se procedió a comprobar la consistencia (v.g. si una clase podría tener instancias) y la clasificación (v.g. inferir un nuevo árbol jerárquico de clase a partir de las definiciones definidas) de los conceptos definidos. Además, el OWL Plugin proporciona un mecanismo con el objeto de comprobar un conjunto de buenas prácticas de diseño de ontologías y si la ontología descrita es conforme con OWL-DL (v.g. avisar al usuario que se han utilizado características de OWL Full como metaclases).

En la figura 6.8 se muestra la captura de la pantalla del editor Protégé en la que se recogen tanto los conceptos y relaciones de la ontología definida (marco superior de la pantalla) y las violaciones de consistencias encontradas durante la elaboración de dicha ontología (marco inferior de la pantalla).

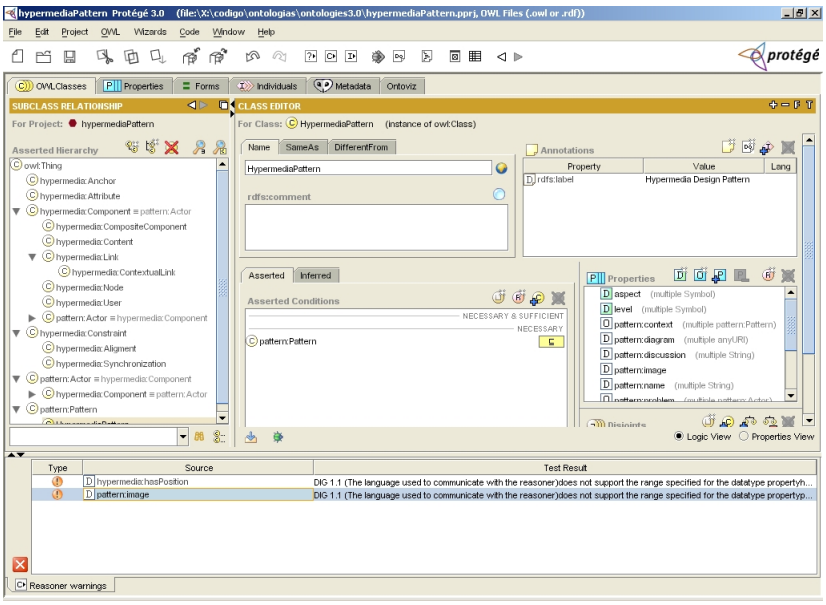


Figura 6.8: Pantalla del editor Protégé mostrando la ontología y la violación de consistencias

Después de haber validado el desarrollo de dichas ontologías, éstas fueron almacenadas en ficheros como conceptos y relaciones de OWL, los cuales han sido recogidos en el anexo B y publicados en un servidor web para que puedan ser utilizados por terceros, como pueda ser el caso de un repositorio web de patrones de diseño hipermedia (véase la sección 6.3.1), de una herramienta para la aplicación de patrones a partir de un conjunto de requisitos (véase la sección 6.3) o la herramienta AnnotPat, la cual fue descrita en la sección 5.3.3, y que proporcionó el soporte tecnológico para llevar a cabo la validación epistemológica que analiza la capacidad expresiva del modelo. Como ya se ha descrito, AnnotPat permite al diseñador o al autor de patrones representar formalmente los patrones de diseño hipermedia recogidos en el anexo A a través de anotaciones semánticas basadas en la ontología de un patrón de diseño hipermedia. En la figura 6.9 se muestra una captura de pantalla de la herramienta AnnotPat tomada durante el proceso de validación, en la que se visualizan un conjunto de patrones anotados que formarán parte de un repositorio.

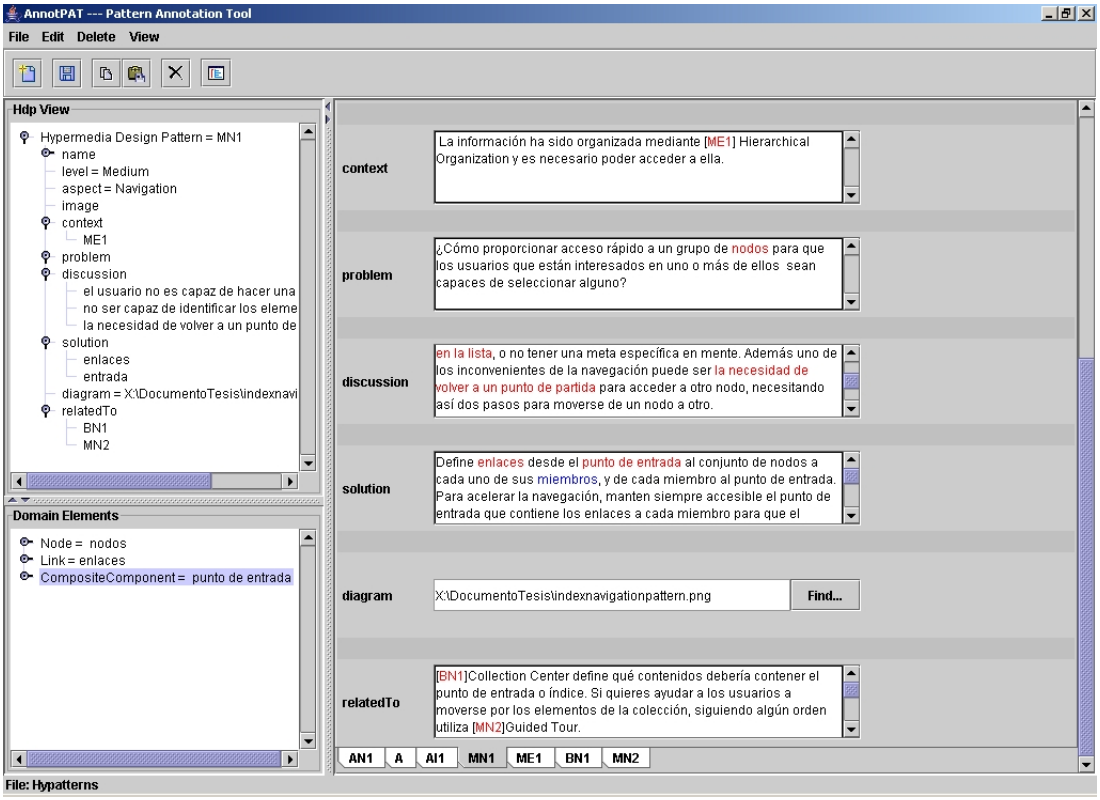


Figura 6.9: Pantalla de AnnotPat durante el proceso de formalización del lenguaje de pdh

En el anexo A se recoge junto con cada uno de los patrones, en el campo denominado Formalización, su descripción formal usando el lenguaje RDF que, al igual que OWL, es estándar para la web semántica y permite describir los datos de una base de conocimiento.

6.2.3. Evaluación de utilidad

Uno de los requisitos para la definición del modelo de representación era que pudiese ser utilizado por los propios usuarios de patrones. Con este objeto se aplicó el mecanismo de anotaciones semánticas, que está siendo utilizado en la actualidad en la web semántica para dotar de significado a los recursos web. Para dar soporte al proceso de anotación se implementó una herramienta que ya ha sido presentada en la sección 5.3.3 y se ha utilizado para realizar la validación epistemológica del modelo de representación mediante la forma-

lización de los patrones recogidos en el anexo A. Sin embargo, para comprobar la verdadera utilidad de la herramienta era necesario realizar una evaluación con usuarios reales.

La evaluación de utilidad ha sido llevada a cabo con dos tipos diferentes de usuarios: usuarios expertos en patrones de diseño hipermedia (cinco de los evaluadores de la sección 6.1.4) y usuarios noveles (otros cinco evaluadores), pero ninguno de ellos tenían conocimiento sobre ontologías. Durante el proceso de evaluación, los evaluadores debían realizar un conjunto de tareas con la herramienta y posteriormente rellenar el cuestionario recogido en la sección C.2. Esas tareas tenían como objeto la creación de un repositorio con un par de patrones, los cuales debían anotar y enlazar. Posteriormente, para comprobar que el repositorio creado con sus propias anotaciones podía ser tratado por una herramienta software, dicho repositorio era exportado a un servidor web para su posterior visualización, navegación y búsqueda, el cual se verá con más detalle en la sección 6.3.1. Para finalizar, los evaluadores tenían como tarea adicional el intentar formalizar un patrón de diseño hipermedia utilizando las propias ontologías que utiliza la herramienta AnnotPat para dar soporte al proceso de anotación, pero en un editor de ontologías como es Protégé. El objeto de esta última tarea era mostrar al evaluador cual sería el modo de formalizar un patrón sin la existencia de AnnotPat.

El cuestionario de evaluación comenzaba con la opinión de los evaluadores sobre la interfaz de la herramienta, acorde a una serie de parámetros sobre su utilidad, rapidez, fiabilidad, etc., y conocer así su grado de usabilidad. En la figura 6.10 puede observarse que AnnotPat tiene un grado de aceptación de “bueno” por parte de sus evaluadores.

El grado de satisfacción global con respecto a la herramienta se puede ver en la figura 6.11, resultando que un 67 % de los evaluadores estaban “muy satisfechos” con el uso de la herramienta. Algunos comentarios de los evaluadores resaltaron la sencillez de navegación entre los patrones y su clara visualización.

El resto del cuestionario profundizaba en cada una de las tareas que tuvieron que realizar los evaluadores. Cabe mencionar tan sólo la tarea relacionada con crear una instancia de un patrón con un editor de ontologías. En la figura 6.12 puede verse cómo se puede crear una instancia mediante la definición previa de cada una de las partes que van a formar ese individuo, es decir, primero se deben definir los elementos que forman parte del problema, para a continuación poder definir el problema, y así sucesivamente hasta poder completar la definición de un patrón. Ante este modo de realizar la formalización de patrones, los evaluadores expresaron su preferencia por AnnotPat, puesto que aunque encontraron a Protégé un

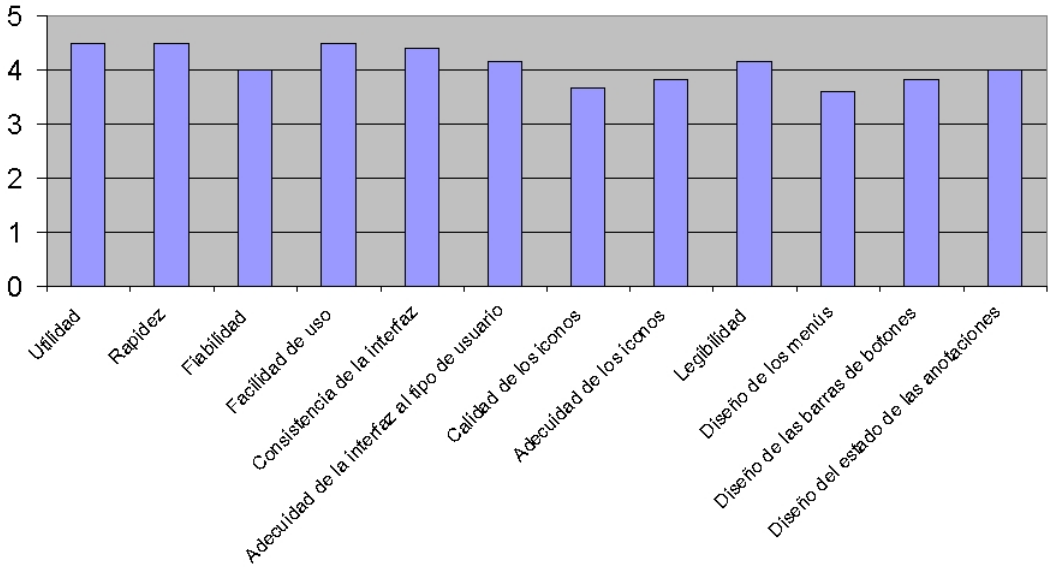


Figura 6.10: Análisis sobre las características generales de AnnotPat

editor muy completo, era menos intuitivo en la construcción de un patrón y no consideraban la interfaz adecuada al tipo de usuario, usuarios noveles en el área de las ontologías.

Para finalizar, hay que comentar algunas mejoras interesantes que propusieron los evaluadores para incluir en una nueva versión de la herramienta:

- Borrar anotaciones desde los árboles de estado.
- Poder modificar el orden de los campos del patrón una vez se han creado.
- Cerrar los patrones desde el menú contextual de su pestaña.
- Mostrar información de asentimiento cuando se ha creado un repositorio.
- Poner las pestañas de los patrones en la parte superior de la ventana.

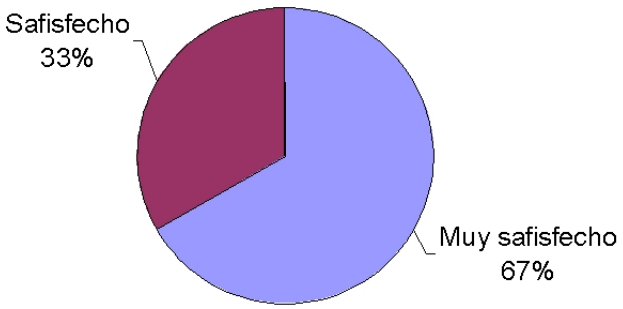


Figura 6.11: Análisis sobre el grado de satisfacción global sobre AnnotPat

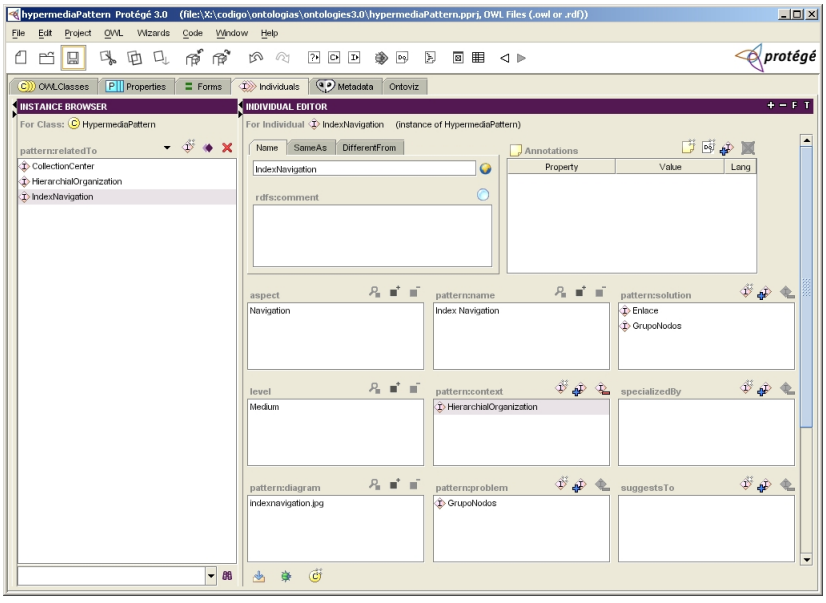


Figura 6.12: Pantalla de Protégé durante la creación de una instancia de patrón

6.3. Evaluación de factibilidad técnica

Para concluir con este conjunto de evaluaciones sobre el trabajo aquí propuesto, se ha procedido a comprobar su factibilidad técnica mediante la creación de dos herramientas que combinan las soluciones propuestas en el nivel cognitivo con el nivel conceptual. La primera de ellas es la construcción de un repositorio web de patrones (véase la sección 6.3.1), a partir del repositorio creado con la herramienta AnnotPat durante la validación epistemológica del modelo de representación en la sección 6.2.2, el cual representa al lenguaje de patrones definido en la sección 4.3. La segunda herramienta es un módulo que automatiza las tareas de refinamiento y adaptación de patrones durante el proceso guiado por patrones durante el paso entre la fase de análisis y la fase conceptual del diseño de las aplicaciones hipermedia. Dicho módulo se ha desarrollado para una herramienta de soporte a un método de diseño concreto (véase la sección 6.3.2).

6.3.1. Un repositorio web semántico para pdh

Para difundir los patrones de diseño hipermedia entre los usuarios y los agentes software se han aprovechado las ventajas de la tecnología de la web semántica subyacentes a la decisión tomada sobre el lenguaje de representación para las ontologías definidas, OWL, que ha permitido desarrollar un repositorio web semántico. El objetivo de este repositorio es recoger y organizar los patrones de diseño hipermedia de acuerdo con el esquema de categorización propuesto en el capítulo 4 para que puedan ser explorados por los usuarios a través de las relaciones recogidas en el lenguaje de patrones. Además, se incluye una herramienta de búsqueda que permite definir varios parámetros, relacionados todos ellos con el conocimiento expresado en el modelo, y poder dirigir así el sentido de la búsqueda.

Detalles de implementación

El repositorio se compone de una serie de módulos orientados a la navegación y consulta de los pdh formalizados según el modelo de representación definido. La arquitectura general se muestra en la figura 6.13.

La ontología de patrones hipermedia se encuentra almacenada como un conjunto de documentos OWL, accesibles vía web en el propio servidor (véase el anexo B). El repositorio

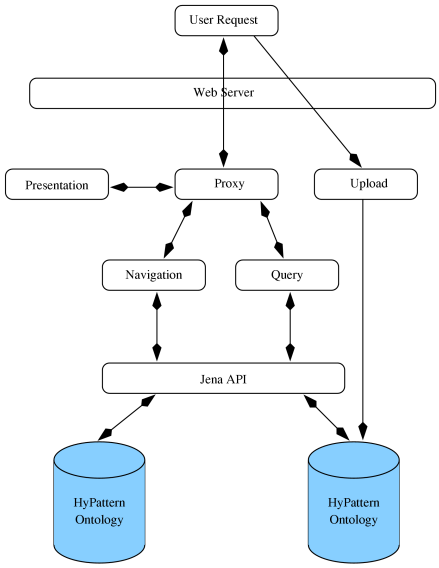


Figura 6.13: Arquitectura del repositorio web

de patrones hipermedia se encuentra en un directorio bajo la estructura del servidor Web y cada instancia de patrón hipermedia se almacena como un fichero RDF con la información propia del patrón y un fichero XML con las anotaciones generadas por AnnotPat.

El acceso a ambos tipos de información se realiza mediante la herramienta de web semántica Jena [22], que permite la manipulación de la ontología OWL y la realización de consultas basadas en RDF empleando RDQL, un lenguaje de consultas propio, y además de la inferencia basada en la ontología.

Desde la interfaz de usuario se puede acceder a la información almacenada en el repositorio empleando la aproximación de navegación por consulta, p.e. nuevos hechos son afirmados mediante la exploración de los hiperenlaces o por búsqueda directa de información. La petición es recibida por un servidor web Apache Tomcat y gestionada por un **módulo proxy**. Este módulo es responsable de la coordinación con el resto de módulos y de responder a las peticiones del usuario. El **módulo de navegación** se encarga de obtener las partes de la ontología o las instancias de acuerdo con el significado del hiperenlace seleccionado, así como de los datos inferidos. Además, tiene que determinar las siguientes relaciones entre el recurso solicitado y los restantes recursos para añadir información a modo de hiperenlaces para las futuras selecciones. El **módulo de consulta** realiza tanto las peticiones realizadas por el módulo de navegación para resolver el destino del enlace como

las consultas realizadas por los propios usuarios. Estas dos acciones son llevadas a cabo con la ayuda de la herramienta Jena y el resultado es formateado semi-automáticamente como páginas web HTML por el **módulo de presentación**, que hace uso de una hoja de transformación XSLT para presentar la información al usuario de manera adecuada.

Ejemplos de navegación

El sitio web permite explorar los patrones de diseño hipermedia mediante la navegación, además de permitir realizar consultas sobre el repositorio de forma que se obtengan los patrones hipermedia que cumplan un conjunto de condiciones.

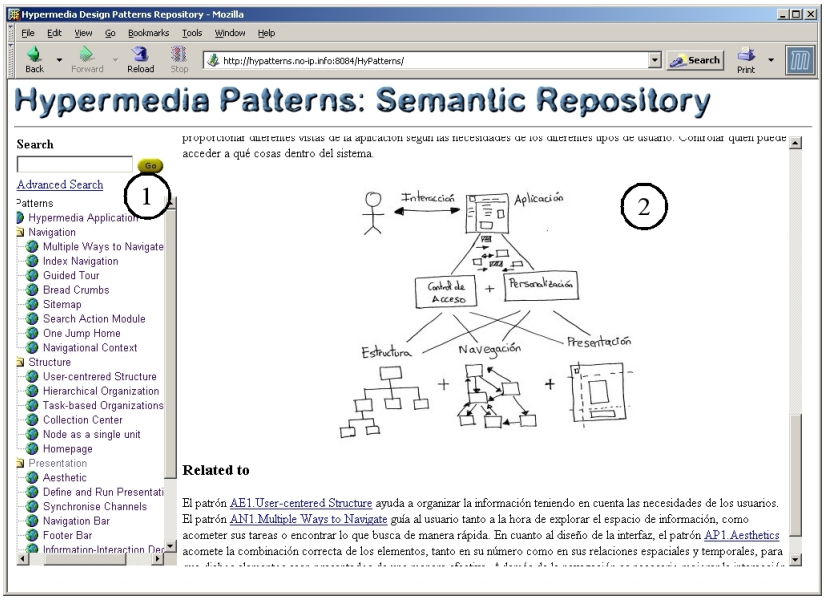


Figura 6.14: Interfaz del repositorio web

La figura 6.14 ilustra cómo se muestra la información de un patrón hipermedia concreto al usuario. La interfaz se divide en dos zonas, la parte izquierda que funciona como herramienta de acceso al lenguaje de patrones y la parte derecha en la que se muestra el contenido de un patrón determinado. La herramienta de acceso se presenta en modo de árbol en el cual los patrones están organizados por aspecto de diseño (estructura, navegación, presentación, interacción, personalización y seguridad) y dentro de cada uno de ellos se encuentra el acceso a los patrones clasificados con esa categoría y ordenados de mayor a menor nivel de descripción. El usuario puede interactuar con dicho árbol para recuperar un

patrón determinado. Cada patrón, en sus campos "Contexto" y "Relacionado con" contiene enlaces semánticos a los diferentes patrones con los que está relacionado. La selección de cada uno de estos enlaces provoca la recuperación del patrón destino, permitiendo así al usuario explorar el espacio de patrones.

La figura 6.15 muestra cómo realizar una consulta por diferentes campos que permita dirigir el resultado de dicha tarea (1). La selección de campos a consultar se va realizando de manera dinámica. Un selector propone un conjunto de criterios, del cual el usuario debe seleccionar uno y escribir la palabra o palabras claves para las cuales quiera conseguir coincidencia. Para añadir nuevos campos a la consulta, el usuario tiene disponible un botón "Add condition" cuya reacción es la generación de un nuevo campo a consulta. De igual manera, puede reducir el número de condiciones mediante el botón "Remove condition". Por ejemplo, la consulta sobre los campos "level=Medium" y "aspect=Presentation" devuelve aquellos patrones cuya clasificación coincida con los valores especificados (2). Cada una de las coincidencias es un enlace cuya selección provoca la recuperación y presentación del patrón, pudiendo hacer así una exploración del espacio de patrones dirigida.

El ejemplo de dicha consulta en lenguaje RDQL sería:

```
SELECT ?name ?pattern WHERE (?pattern pattern:name  ?name)
(?pattern <http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl#level> "Medium")
(?pattern <http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl#aspect> "Presentation")
USING pattern FOR <http://hypatterns.no-ip.info/ontologies/pattern.owl#>
```

6.3.2. Módulo para la aplicación semi-automática de pdh en una herramienta de diseño

Para llevar a cabo esta evaluación se ha optado por la integración del módulo en una herramienta de diseño de la cual se disponía el código fuente y cuyo desarrollo se ha realizado dentro del grupo de investigación DEI del Departamento de Informática de la Universidad Carlos III de Madrid. Esta herramienta, llamada AriadneTool [Montero *et al.*, 2004a], está basada en uno de los métodos presentados en la sección 2.2.3, ADM [Díaz *et al.*, To be published]. Otra de las características que permitía una mejor integración era contar con la representación del modelo de referencia del método, Labyrinth [Díaz *et al.*, 1997, Díaz *et al.*, 2001b], en un lenguaje de representación de ontología [Montero *et al.*, 2003c], lo que ha permitido comprobar la interoperabilidad del modelo de representación definido en esta tesis con respecto a otra representación que conceptualiza

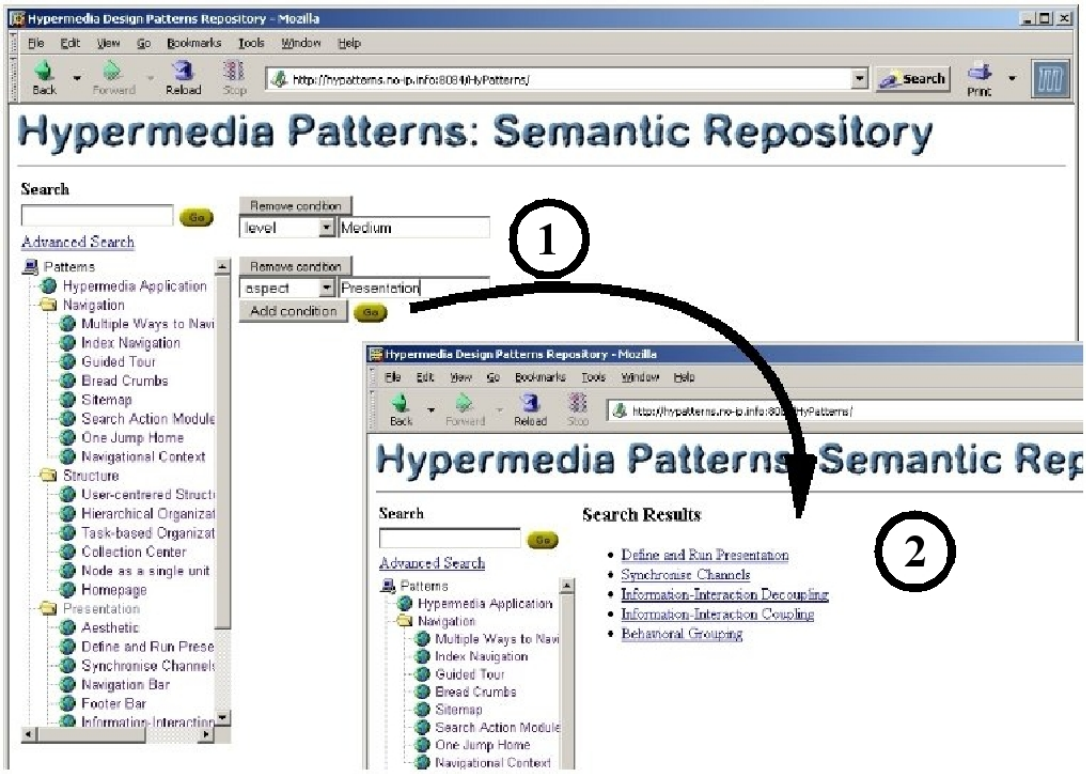


Figura 6.15: Ejemplo de resultados de una búsqueda en el repositorio web

el mismo dominio. También hay que mencionar que existe un área importante dentro del dominio de las ontologías dedicada a la definición de heurísticas de transformación entre ontologías (*ontology mapping* u *ontology alignment*), el cual queda fuera del alcance de esta tesis. La traducción de ambas ontologías se ha realizado de manera artesanal utilizando los axiomas de equivalencia a nivel de concepto y de relación.

AriadneTool es un entorno orientado al desarrollo de aplicaciones hipermedia y, actualmente, automatiza gran parte del proceso de diseño de ADM y soporta el prototipado rápido en los lenguajes de marcado HTML, XML, SMIL y RDF, así como la generación automática de la documentación del proceso de diseño. Además, AriadneTool también incorpora el uso de ontologías para verificar la completitud, consistencia y corrección del diseño con respecto de la semántica de ADM y en consecuencia mejora la comprensión del usuario en su utilización.

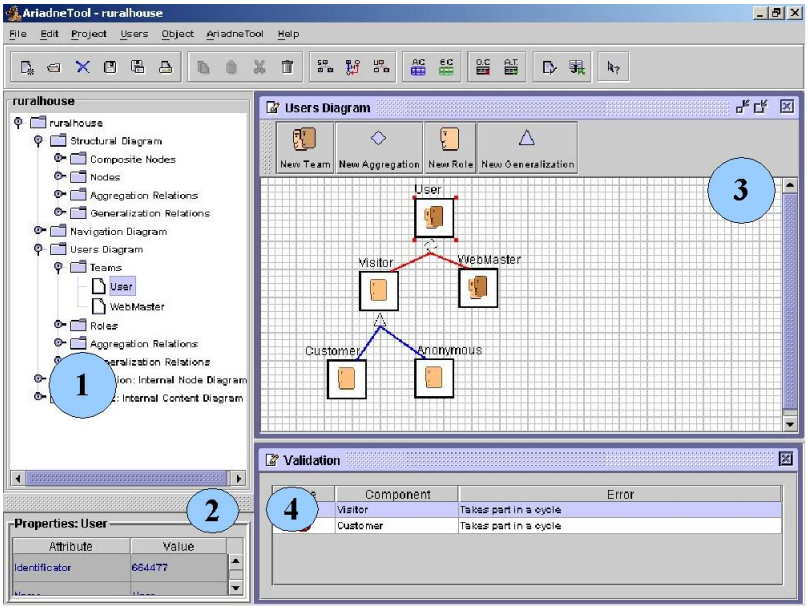


Figura 6.16: Interfaz gráfico de AriadneTool

Cada sistema hipermedia o web desarrollado con la herramienta se corresponde con un proyecto. La figura 6.16 muestra la interfaz gráfica de usuario para la gestión de un proyecto, la cual se encuentra dividida en cuatro áreas principales:

- El **navegador del proyecto**, desde el que puede accederse a los diagramas y elementos del modelo que se está realizando, empleando una estructura de árbol (área 1 de la figura 6.16).
- El **panel de propiedades**, en el que se muestran los atributos del elemento del modelo o diagrama seleccionado (área 2).
- El **panel de edición**, en el que se muestran y editan los diferentes diagramas del método ADM (área 3).
- El **panel de validación**, en el que se muestran al usuario los mensajes de error y los avisos referentes a las violaciones de la semántica del método (área 4).

Otras funcionalidades de AriadneTool son los menús contextuales, las barras de herramientas, la ayuda que proporciona asistencia inmediata a los usuario sin abandonar el contexto en el que están trabajando, y la generación de código.

Detalles de implementación

El módulo para la aplicación semi-automática de los patrones de diseño hipermedia a partir de la especificación de requisitos se ha integrado en la arquitectura de AriadneTool como se muestra en la figura 6.17.

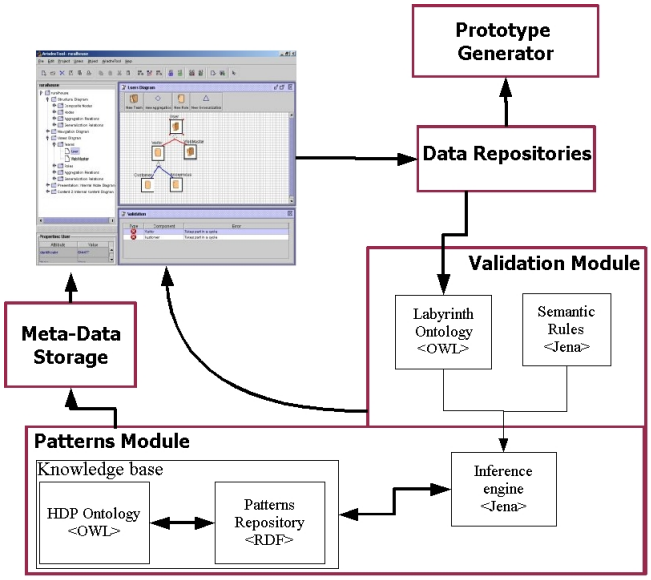


Figura 6.17: Arquitectura de AriadneTool

El **módulo de patrones** se encarga de gestionar la base de conocimiento de los patrones y de llevar a cabo la semi-automatización del proceso de aplicación de dichos patrones a partir de un conjunto de requisitos, que se mostrará más adelante. Este módulo comparte con el **módulo de validación** la herramienta Jena para el soporte a la gestión de la base de conocimiento y el sistema de inferencia para la selección y aplicación de los patrones. Además necesita del **almacén de meta-datos** para poder representar un patrón instanciado según las entidades de diseño en el producto correspondiente de ADM. Adicionalmente, en el **repositorio de datos** se almacenará la solución resultante del proceso de aplicación de patrones, en forma de proyecto para que pueda ser gestionado por AriadneTool. Finalmente, el **generador de prototipos** puede producir un prototipo de la aplicación resultante, después de completar el diseño con detalles más específicos (v.g. identificar qué recursos corresponden a cada contenido definido).

Ejemplo de aplicación

Retomando el caso de estudio que se describió en la sección 6.1.2 sobre cómo se puede afrontar el paso entre la fase de análisis y la fase de diseño conceptual a través de un proceso guiado por patrones, queda como último paso la instanciación de los esquemas de solución propuestos a entidades de diseño propias del método que se esté utilizado para llevar a cabo el proceso de diseño. En este caso se utilizará la herramienta AriadneTool para completarlo.

El proceso se desencadena a partir de la creación de un proyecto nuevo en la herramienta y la selección en el menú "Project" de la opción "Requirements Information". Como primer paso ya conocido, es la especificación de los requisitos, en este caso a través de la herramienta. La figura 6.18 muestra la pantalla que se encarga de esa recogida de requisitos, y para la cual se contará con una plantilla a rellenar adaptada de [Preece *et al.*, 2002]. Al final de la plantilla hay un selector para llevar a cabo la selección de aquel patrón que mejor cubra el requisito especificado.

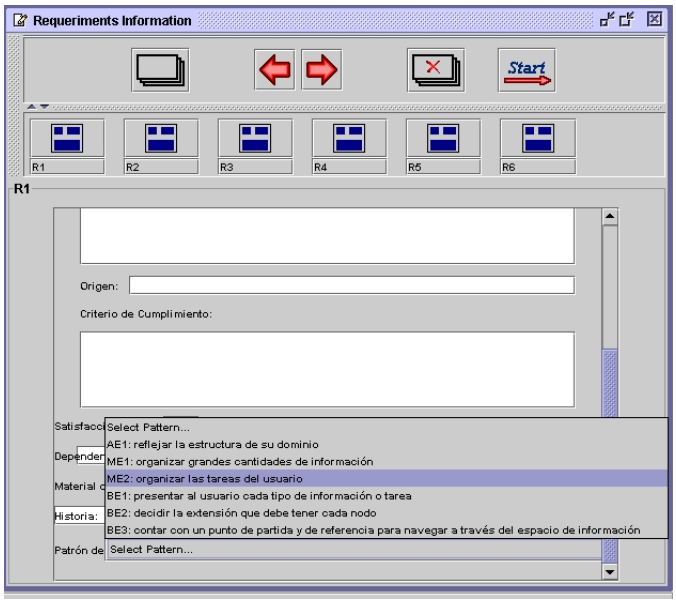


Figura 6.18: Pantalla de recogida de requisitos en AriadneTool

Una vez introducido los requisitos y seleccionados los patrones se puede comenzar con el proceso de refinamiento de los mismos, accionando el botón *"Start"*. Recordemos, que este proceso sólo se iniciará si los patrones seleccionados pertenecen a la categoría de nivel medio o bajo, de lo contrario la aplicación avisará al usuario para realizar las modificaciones

pertinentes. En la figura 6.19 muestra el asistente que guiará el proceso de transformación desde los requisitos a un modelo de diseño conceptual a través de las soluciones de los patrones (Paso 3, Paso 4 y Paso 5 del proceso). Como primer paso en el asistente aparece una tabla resumen con los requisitos y su patrón asociado. El icono con figura de lupa que aparece en cada una de las filas de la tabla permite la visualización del patrón a través del repositorio web mediante su pulsación.

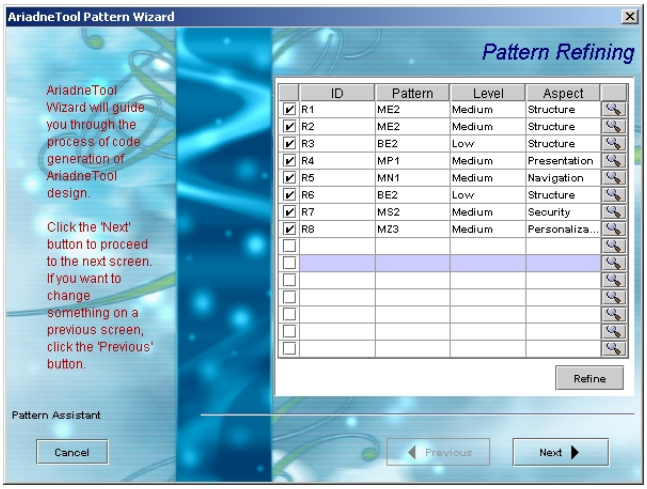


Figura 6.19: Pantalla del asistente que guía el proceso basado en patrones en AriadneTool

A continuación, se pueden refinar esos patrones seleccionados mediante las relaciones existentes en el lenguaje de patrones, pulsando el botón *"Refine"*. En la figura 6.20 puede verse los patrones adicionales que propone el asistente a los ya existentes.

Una vez que se ha finalizado con el refinamiento de los patrones y todos los seleccionados están catalogados con nivel medio o bajo, se puede continuar con el siguiente paso pulsando en el asistente al botón "Next". La figura 6.21 muestra el proceso de instanciación de los patrones. En la parte izquierda del asistente aparecen organizados los requisitos según el aspecto de diseño al que pertenecen, como preparación a la entrada en el diseño conceptual. Cada requisito tiene a su vez los patrones seleccionados para resolver el problema que plantean. La selección de un determinado patrón provoca la creación de una plantilla con la solución de diseño propuesta por dicho patrón para que el usuario cree tantas instancias de los elementos de diseño involucrados como necesite. La figura 6.21 muestra la solución del patrón `[MZ1]Role Subtype` que permite describir la jerarquía de roles para el sistema, en

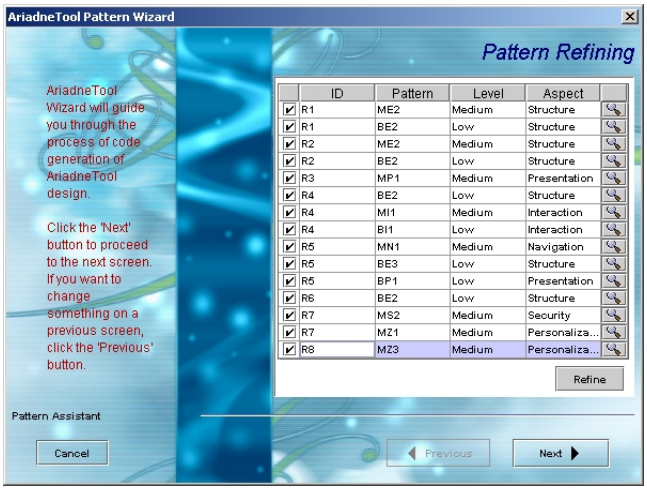


Figura 6.20: Pantalla que muestra los patrones seleccionados tras la consulta al repositorio

este caso son los roles especificados en el requisito R7 del sistema arce (véase la sección 6.1.2).

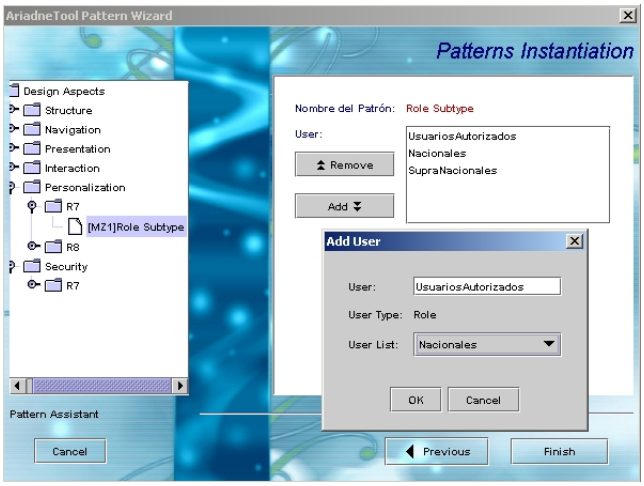


Figura 6.21: Pantalla que muestra la instanciación de un patrón de diseño en AriadneTool

Una vez creadas las instancias para cada uno de los patrones se puede proceder a generar los diferentes productos del modelado conceptual, pulsando el botón *"Finish"*. Esta acción provoca el paso de las instancias creadas a elementos de diseño propios del método y la elaboración de cada uno de los productos correspondientes.

Siguiendo con el ejemplo del sistema Arce, la figura 6.22 muestra el resultado de la instanciación del patrón `[MZ1]Role Subtype` en el Diagrama de Usuarios del método ADM. En dicha figura puede verse como se han creado tipos de entidades roles para cada uno de los tipos de usuarios especificados en el requisito R7 y se han organizado según la jerarquía especificada.

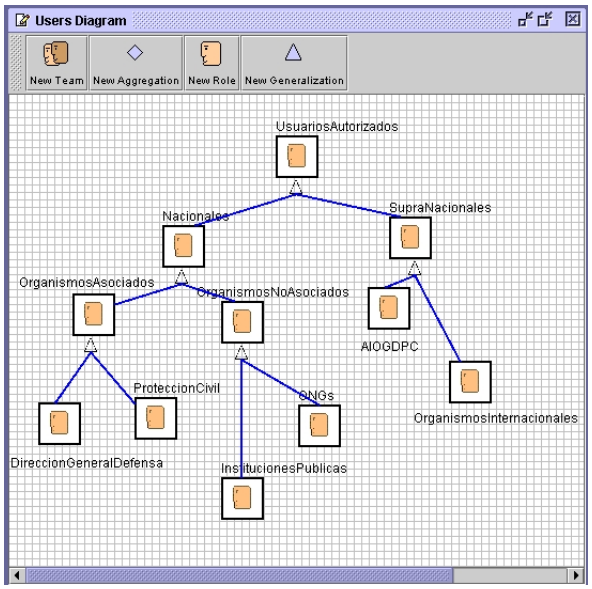


Figura 6.22: Diagrama de usuarios del sistema Arce

La figura 6.23 muestra el resultado de la instanciación de los patrones de diseño estructurales como producto de ADM en el Diagrama Estructural. En este diagrama se recogen todas las instancias definidas como CompositeComponent y Node manteniendo sus relaciones de agregación y especialización.

La figura 6.24 muestra el Diagrama de Navegación de ADM como resultado de la adaptación del patrón `[MN1]Index Navigation`. En este diagrama se recogen todas las instancias definidas como Link y los nodos con los que están relacionados.

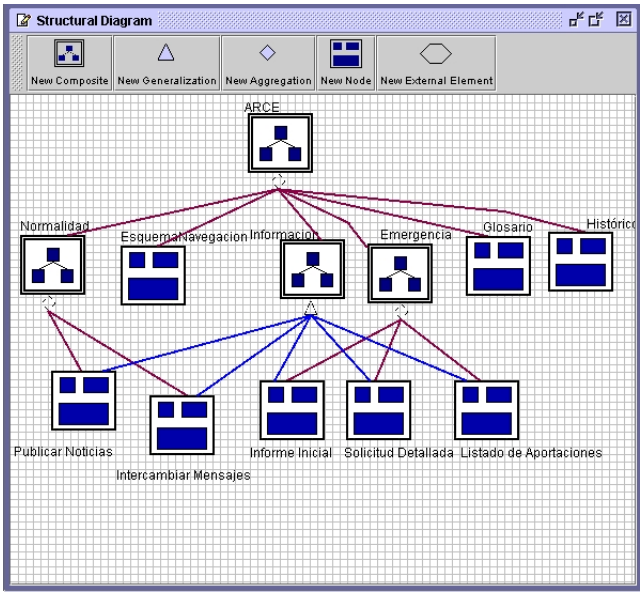


Figura 6.23: Diagrama estructural del sistema Arce

La figura 6.25 muestra el Diagrama Interno del nodo “Información” en el cual se aplica el patrón `[MP1]Information-Interaction Decoupling`. En este diagrama se recogen todas las instancias definidas como Content y Anchor.

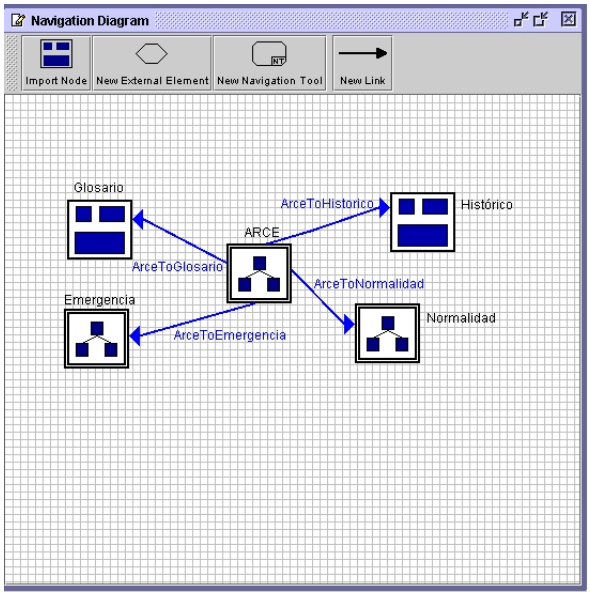


Figura 6.24: Diagrama de navegación del sistema Arce

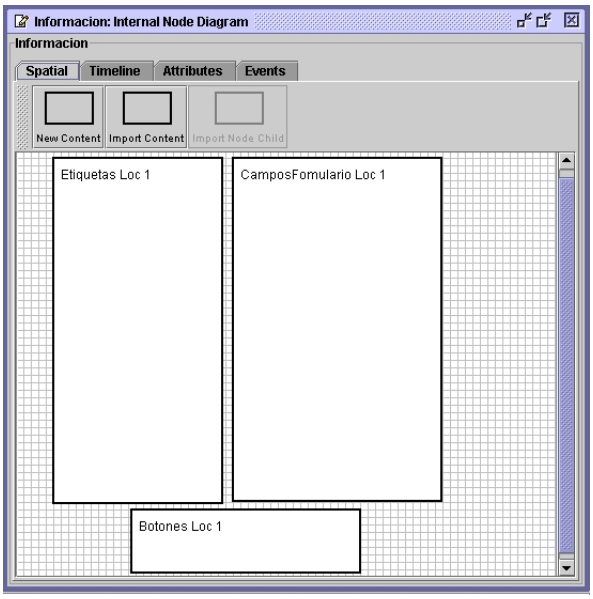


Figura 6.25: Diagrama de presentación de un nodo del sistema Arce

El resto de aspectos de diseño, interacción, personalización y seguridad, tenían como adaptación la inclusión eventos en el producto Especificación de funciones y un regla de acceso en el producto Tabla de Acceso de ADM. Para ver una mayor descripción del significado de los elementos de diseño presentados en las figuras anteriores se puede consultar [Montero *et al.*, 2004a].

6.4. Resumen del capítulo

En este capítulo se han mostrado el conjunto de evaluaciones llevadas a cabo para determinar tanto la completitud como la factibilidad técnica y utilidad de la solución propuesta.

A continuación, se muestra un resumen de dichas evaluaciones junto con su objetivo, los elementos que han participado y el resultado de realizar dicha evaluación.

Validación de la completitud del lenguaje de pdh

- **Objetivo:** Determinar si el lenguaje aporta las relaciones necesarias para guiar al diseñador sobre cómo combinar un conjunto de patrones para resolver un problema de diseño complejo, el diseño de una aplicación hipermedia.
- **Participantes:** El lenguaje de patrones.
- **Resultado:** El conjunto de patrones seleccionados (catálogo) tiene el estatus de lenguaje de patrones según [Todd *et al.*, 2004], gracias a la integridad demostrada en las relaciones definidas entre los patrones, así como al espacio de categorización bidimensional propuesto, que permite organizar los patrones en tres niveles diferentes de abstracción (alto, medio y bajo) cubriendo cada uno de ellos los diferentes aspectos de diseño de una aplicación hipermedia (estructura, navegación, presentación, interacción, personalización y seguridad). Además, el número de relaciones de "asociación" definidas en el lenguaje permiten al diseñador ampliar el contexto del problema y confieren al lenguaje un alto grado de madurez.

Evaluación de factibilidad del lenguaje de patrones

- **Objetivo:** Determinar si el lenguaje puede ser utilizado para generar diseños.

- **Participantes:** El lenguaje de patrones y el proceso de transformación de requisitos a un diseño conceptual guiado por patrones.
- **Resultado:** Los pasos definidos para el proceso de integración de los patrones en el proceso de diseño hipermedia han permitido guiar la transformación de un conjunto de requisitos de un caso real, como es el sistema Arce [10], a su especificación como un conjunto de esquemas para cada uno de los aspectos de diseño hipermedia que conforman una primera aproximación al modelado conceptual a través de las soluciones propuestas por los patrones del lenguaje.

Evaluación analítica para la integración con los métodos

- **Objetivo:** Determinar la capacidad de los métodos de diseño para recibir los esquemas resultantes del proceso de integración.
- **Participantes:** Los métodos de diseño.
- **Resultado:** De los métodos analizados, RMM, OOHDM, WSDM, WebML, OO-H y ADM, tan sólo este último podría recibir una entrada ordenada de los esquemas resultantes del proceso de integración. Con respecto al resto de métodos serían los problemas de diseño relacionados con la seguridad del sistema los que no tendrían una solución en su modelado conceptual.

Evaluación de utilidad para el proceso de integración

- **Objetivo:** Determinar el grado de satisfacción del marco de integración de los pdh en el proceso de diseño y su posibles dificultades en su aplicación.
- **Participantes:** Evaluadores experimentados, el lenguaje de patrones y el proceso de integración.
- **Resultado:** Los evaluadores tras aplicar el proceso de integración a un caso de uso determinaron su satisfacción en general con la utilidad del proceso, con el modelado resultante y con la organización del lenguaje de patrones. Con respecto a posibles dificultades, encontraron problemas a la hora de detectar el patrón que cubría requisitos demasiado concretos, ya que esto suponía una mayor profundización en la descripción y conocimientos sobre el lenguaje de patrones.

Evaluación analítica del modelo de representación

- **Objetivo:** Determinar el grado de cumplimiento con respecto a los requisitos de una representación formal para pdh.
- **Participantes:** Modelo de representación y otras aproximaciones existentes.
- **Resultado:** El modelo de representación cumple los requisitos exigibles y sus puntos fuertes con respecto al resto de aproximaciones son formalizar todos los campos del patrón lo que permite describir un lenguaje de patrones de manera completa, como desarrollar aplicaciones para la exploración de patrones, salvaguardando así la cualidad comunicativa de los mismos.

Validación formal y epistemológica del modelo

- **Objetivo:** Determinar la validez formal y la capacidad expresiva del modelo de representación.
- **Participantes:** El modelo de representación, un editor de ontologías y un sistema de inferencia.
- **Resultado:** La especificación del modelo de representación ha pasado la batería de test de consistencia, de clasificación y de buenas prácticas de diseño de ontologías que fueron ejecutadas por el editor. Además, el modelo de representación ha probado su capacidad expresiva al describir formalmente los patrones recogidos en el anexo A en el campo "Formalización".

Evaluación de utilidad de AnnotPat

- **Objetivo:** Determinar el grado de utilidad y satisfacción de la herramienta para la formalización de patrones.
- **Participantes:** Evaluadores expertos y noveles y AnnotPat.
- **Resultado:** Los evaluadores determinaron su satisfacción con la herramienta de soporte al proceso de anotación de patrones, resaltando su sencillez con respecto a la tarea de formalizar un patrón en un editor de ontologías, a la vez que propusieron una serie de mejoras a la interfaz de la herramienta.

Evaluación de factibilidad técnica

- **Objetivo:** Determinar si a partir del nivel conceptual se podía automatizar el nivel cognitivo de la solución propuesta.

- **Participantes:** El modelo de representación, el lenguaje de patrones y el proceso de integración de los patrones en el proceso de diseño.
- **Resultado:** Se han desarrollado dos herramientas con objetivos diferentes. La primera de ellas, un repositorio web, permite la exploración y búsqueda sobre el lenguaje de patrones. La segunda de ellas automatiza el proceso de transformación de los requisitos a un modelo conceptual guiado por patrones en una herramienta de diseño concreta. Ambas herramientas comparten un repositorio que representa al lenguaje de patrones recogido en el anexo A formalizado según el modelo de representación propuesto.

En el siguiente capítulo se procederá a comprobar que la solución propuesta cubre los objetivos y requisitos especificados al comienzo de este documento.

Capítulo 7

Conclusiones

La finalidad de esta tesis es hacer una aportación significativa al área de la Ingeniería Hipermedia, y más concretamente, en la adopción de los patrones de diseño hipermedia en el proceso de desarrollo de las aplicaciones, dando un paso más hacia su práctica habitual.

Para conseguir este objetivo, se ha llevado a cabo una organización y conceptualización del espacio de los patrones, además de haberse elaborado un marco de integración de los mismos en el proceso de diseño hipermedia, lo que en su conjunto ha proporcionando una base para el desarrollo de aplicaciones que ayudan a la gestión de repositorios de patrones, su exploración e integración con herramientas de soporte al diseño.

En las siguientes secciones se presentan las conclusiones obtenidas de la investigación y del trabajo realizado en esta tesis (véase la sección 7.1). Después, se describen las aportaciones originales que han surgido de la elaboración de este trabajo (véase la sección 7.2). Finalmente, se presentan las líneas de investigación que quedan abiertas como continuación a este trabajo (véase la sección 7.3).

7.1. Conclusiones

La metáfora de patrón es independiente del dominio y su genialidad radica en su modelo descriptivo, que permite organizar el conocimiento, registrando nuestra experiencia de tal manera que no pueda ser olvidada y que aquellos fallos o errores que cometimos no se repitan.

A parte de este hecho, la elaboración de este trabajo y la realización de su proceso de evaluación han permitido extraer una serie de conclusiones propias sobre el papel de los patrones en el proceso de diseño:

- En primer lugar, su utilidad irrefutable para capturar aquellos problemas que se repiten en el dominio de la aplicación, y reutilizarlos en contextos diferentes. En concreto, en el dominio de las aplicaciones hipermedia y web su utilidad radica en aportar buenas prácticas de diseño al caos impuesto en la forma de desarrollo de estas aplicaciones, siendo prueba de ello la propia web actual.
- Sin embargo, esta utilidad lleva aparejada la dificultad que entraña el uso de patrones. El diseñador debe enfrentarse a una solución fruto de la experiencia de otros diseñadores en la resolución reiterada a un problema, la cual requiere de un proceso cognitivo de asimilación para que el propio diseñador pueda aplicar el patrón a un contexto particular.
- Con el objeto de aliviar esta carga cognitiva, se produce un énfasis en la automatización de su aplicación, olvidando a veces el rol del patrón como herramienta de comunicación.
- Prueba de su valor comunicativo ha sido la utilización de patrones de diseño web e hipermedia como herramienta educativa durante parte de la elaboración de esta tesis, donde los alumnos han aprendido tanto a diseñar interfaces web como a generar diseños conceptuales a partir de la consulta de patrones.
- De esta última experiencia, en la cual los alumnos debían generar sus propios prototipos a partir de un patrón concreto de aplicación de los descritos en [van Duyne *et al.*, 2002], también se ha podido constatar la utilidad de los lenguajes de patrones para refinar e identificar los requisitos de las aplicaciones web.

A parte de estas conclusiones generales, la elaboración de esta tesis ha demostrado la factibilidad de integrar los pdh en el proceso de diseño sin perder ninguna de sus cualidades, tanto en su perspectiva manual como software. Sin embargo, hay que mencionar que se han detectado una serie de carencias ajenas a este trabajo, pero que en algunos casos puede afectar a su generalidad:

- Durante el proceso de construcción del catálogo, algunas de las categorías han sido cubiertas con patrones prestados de otras áreas de conocimiento, como es el caso de

los patrones de seguridad [Fernández y Pan, 2001], que todavía no han sido escritos para el dominio hipermedia, y otras como los niveles bajo de los aspectos de personalización y de seguridad han tenido que ser dejados en blanco. Esto pone de evidencia la necesidad de descubrir nuevos patrones que ayuden a los diseñadores a afrontar el diseño de las aplicaciones hipermedia de manera más completa.

- Además, a partir de la evaluación analítica realizada sobre la adecuación de la aplicación de los patrones a los diferentes productos del modelado conceptual de los métodos de diseño recogidos en la sección 2.2.3 se evidencia que no todos lo métodos pueden aplicar las soluciones de ciertos patrones de diseño, ya que no cuentan con las entidades de diseño pertinentes.

7.2. Aportaciones

La aportación fundamental de esta tesis al área de los patrones de diseño en el dominio de la ingeniería hipermedia se puede resumir en:

- La elaboración de una serie de herramientas que dan soporte a los pdh, tanto en su utilización individual como en su integración con los métodos de diseño.
- La especificación de un modelo de representación semántico que respeta la naturaleza literaria de los patrones y que posibilita el desarrollo de herramientas no sólo para su aplicación, sino para su organización, recuperación y exploración.

Estas aportaciones repercuten en las actividades normales que realiza un diseñador cuando se enfrenta al uso de patrones, bien en la identificación de un patrón o patrones que resuelvan un determinado problema de diseño o en la generación de un modelado conceptual basado en la combinación de una serie de patrones. La tabla 7.1 recoge los problemas que se le pueden plantear al diseñador cuando realiza cada una de estas actividades y qué herramientas ha aportado la elaboración de esta tesis doctoral para aliviar la realización de dichas actividades.

Desde un punto de vista más teórico, la figura 7.1 muestra la correspondencia existente entre cada uno de los objetivos (etiquetados con la letra O) y los requisitos (etiquetados con la letra R) definidos en la sección 1.3. Cada uno de esos requisitos ha sido cubierto con una

Actividad	**Problema**	**Solución**
1. Identificación de un patrón o patrones	1. Recorrer todos los recursos, tanto web como bibliográficos. 2. Intentar recuperar aquellos patrones relevantes basándose en la descripción de su nombre. 3. Discernir a partir de su descripción si es un patrón aplicable.	Espacio de categorización Catálogo de patrones Repositorio web de patrones
2. Adaptación al contexto donde quiere ser aplicado	1. Interpretar el patrón según el método o modelo utilizado en su descripción. 2. Decidir en qué aspecto de diseño tiene que ser aplicado. 3. Adaptar la solución del patrón a las entidades de diseño de un método concreto.	Modelo de presentación formal Proceso de integración en el proceso de diseño de los métodos
3. Identificación del siguiente patrón a aplicar	1. Comprobar si se ha satisfecho la necesidad que originó la aplicación de patrones, en caso contrario 2. Volver a la actividad 1	Lenguaje de patrones Repositorio web de patrones Proceso de integración en el proceso de diseño de los métodos

Tabla 7.1: Resumen de las herramientas desarrolladas para el soporte al uso de patrones

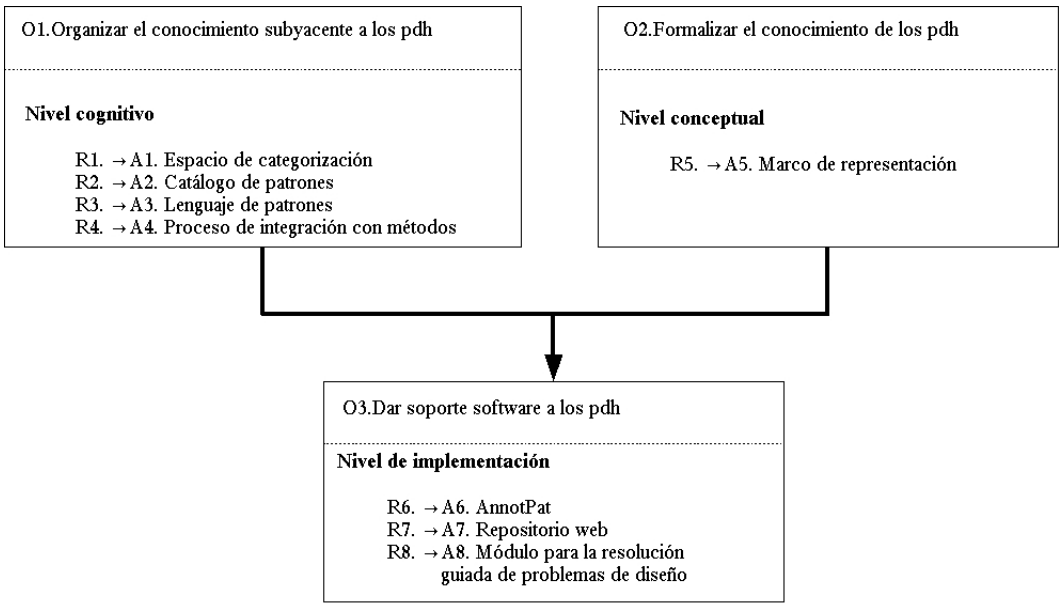


Figura 7.1: Relación entre los objetivos específicos y las aportaciones

determinada aportación (etiquetadas con la letra A). Este trabajo ha elaborado una solución que permite la integración de los pdh en el proceso de diseño basada en tres niveles de abstracción:

Un **nivel cognitivo**, que organiza el conocimiento subyacente a los pdh para ser utilizado tanto de manera individual como en conjunción con los métodos de diseño por los diseñadores hipermedia de manera manual.

Un **nivel conceptual**, que formaliza la estructura y semántica de los pdh proporcionando un vocabulario común a partir de los cuales los patrones de diseño hipermedia han sido descritos independientemente de métodos y modelos concretos.

Un **nivel de implementación** que ha sido fruto de los desarrollos software realizados durante el proceso de evaluación descrito en el capítulo 6 y que permiten la organización, recuperación, exploración y aplicación de los pdh.

A continuación se enumeran las aportaciones concretas de cada nivel.

- El **nivel cognitivo** ha proporcionado un conjunto de mecanismos que dan soporte a los usuarios de los patrones de diseño hipermedia [Montero *et al.*, 2005]. Éstos son:

- Un espacio de categorización que permite clasificar los patrones de diseño hipermedia de acuerdo a la naturaleza del problema en el que serán aplicados, el modelado conceptual de las aplicaciones hipermedia, y a los diferentes niveles de abstracción encontrados en la descripción textual de los patrones.

- Un catálogo de patrones formado por 34 patrones, seleccionados a partir de la bibliografía disponible, que resuelven problemas específicos de diseño de cualquier aplicación hipermedia, clasificados según el espacio de categorización propuesto.

- Un lenguaje de patrones, producto del estudio de las relaciones estructurales existentes entre los patrones seleccionados, a partir de las cuales son organizados y presentados de manera cohesiva por niveles descendentes de abstracción, orientados a un objetivo común: el diseño de una aplicación hipermedia.

- Un proceso de integración con los métodos donde los requisitos guían una aproximación basada en patrones para afrontar el modelado conceptual.

- El **nivel conceptual** proporciona el formalismo necesario para que los patrones puedan ser tratados computacionalmente [Montero *et al.*, 2003a], mediante:

  - Un modelo de representación que define formalmente tanto la estructura como el vocabulario hipermedia utilizado para describir la esencia del patrón. Todo ello es representado mediante ontologías que son utilizadas como formalismo subyacente para complementar la descripción textual del patrón mediante anotaciones semánticas. Dicha solución hace que la formalización del patrón sea transparente al usuario mientras que, a la vez, aporta la rigurosidad necesaria para hacerlos procesables por programas software.

- El **nivel de implementación** proporciona la arquitectura que permite el desarrollo de herramientas para organizar, compartir, explorar y aplicar los pdh. El elemento central de dicha arquitectura es una base de conocimiento formada por el modelo de representación y el lenguaje de patrones anotados semánticamente con la ayuda de una herramienta de anotaciones llamada AnnotPat [Montero *et al.*, 2004b]. A partir de esta base de conocimiento y con la ayuda de un sistema de inferencia se han desarrollado dos casos concretos:

- Un repositorio web semántico como medio para la diseminación del conocimiento de los patrones de diseño hipermedia, que permite la exploración y búsqueda de los patrones.
- La automatización del proceso que guía el paso de la fase de análisis a la fase de diseño conceptual mediante la aplicación de patrones en una herramientas CASE, en particular, la herramienta conocida como AriadneTool [Montero *et al.*, 2004a] que soporta el desarrollo de aplicaciones hipermedia basado en el método ADM [Díaz *et al.*, 2001a, Díaz *et al.*, To be published].

7.3. Líneas de trabajo futuras

A lo largo del desarrollo de esta tesis, así como a la finalización de la misma, se han observado posibles líneas de trabajo futuro, tanto en ampliaciones para el marco de integración con los métodos, el modelo de representación y las herramientas desarrolladas (véase la sección 7.3.1), como varias líneas de investigación complementarias o relacionadas con la propuesta presentada en este trabajo (véase la sección 7.3.2). A continuación se describen las más importantes.

7.3.1. Extensiones al trabajo realizado

En esta sección se indican algunas acciones de ampliación al trabajo realizado, las cuales se estiman interesantes y posibles de abordar a corto y medio plazo.

En primer lugar, el marco de integración de los patrones ha permitido organizar el espacio de los pdh y adecuarlo al proceso de diseño de las aplicaciones hipermedia. Partiendo de este núcleo, nuevas ideas permitirán mejorar y generalizar algunos aspectos del marco propuesto:

- Incluir más patrones de nivel de descripción alto para potenciar el aspecto comunicativo de los patrones por parte de un equipo de desarrollo multidisciplinar.

  El criterio **Nivel de descripción** de los patrones, que definía el nivel de abstracción del patrón, también puede ser visto como los diferentes niveles de conocimiento y habilidades que poseen los diseñadores hipermedia, los cuales, como ya se ha mencionado, forman un grupo heterogéneo donde hay cabida para ingenieros, diseñadores

gráficos, lingüistas e incluso los destinatarios de la aplicación. Por lo tanto, el nivel alto estaría destinado para usuarios o clientes, el nivel medio para diseñadores expertos y el nivel bajo para diseñadores noveles. Así el nivel alto recoge 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, los cuales deberían ser adaptados al dominio de la aplicación para hacerlos más asequibles. Sobre esta línea se está trabajando actualmente en un proyecto de investigación que tiene como objeto una nueva interfaz para el sistema web ARCE [10].

- Estudiar las relaciones de los pdh desde la perspectiva de los requisitos no funcionales y de usabilidad que se encuentran dispersos por los diferentes campos de los patrones y el impacto que puedan tener en la solución propuesta [Gross y Yu, 2000].

En segundo lugar, el modelo de representación semántico ha sido utilizado para la formalización de los patrones de diseño hipermedia teniendo en cuenta la semántica propia de los dominios involucrados. Partiendo de dicho modelo:

- Utilizar las ontologías presentadas para crear nuevo conocimiento sobre los pdh, en concreto, para que las personas que forman parte del equipo de desarrollo de una aplicación hipermedia expresen sus propias experiencias y valores adquiridos durante el desarrollo de un proyecto en forma de patrón.

Finalmente, las herramientas desarrolladas han servido para evaluar la factibilidad técnica de las ideas aquí propuestas, por ello son vistas como prototipos con el suficiente potencial a partir de cual incluir nuevas funcionalidades que redunden en una mayor utilidad. Entre dichas mejoras, destacar:

- Hacer de AnnotPat no sólo un medio para formalizar patrones de una manera transparente, sino un gestor de lenguajes de patrones que dé soporte al usuario o al autor de patrones durante la elaboración y mantenimiento del mismo, como en [Greene *et al.*, 2003], pero que además permita comprobar la integridad y completitud del lenguaje en todo momento mediante la automatización del test realizado en la sección 6.1.1.

- Hacer del repositorio web semántico no sólo un sitio donde visualizar o explorar el lenguaje de patrones desde diferentes perspectivas, sino proporcionar soporte para la comunicación y compartición de los pdh por la comunidad hipermedia.
- Permitir a los usuarios de AriadneTool utilizar los patrones de manera individualizada sin tener que llevar a cabo el proceso dirigido por requisitos.

7.3.2. Perspectivas de trabajos relacionados

Con independencia de las extensiones mencionadas en la sección anterior sobre el trabajo ya realizado, algunas de las cuales se están realizando en la actualidad, existen varias líneas de investigación posibles que complementarían la propuesta presentada.

- Profundizar en el papel de los pdh en las etapas de Análisis y Evaluación.

  Los patrones de diseño pueden tener una variedad de usos. En la elaboración de esta tesis doctoral se ha trabajado con su capacidad para generar diseños, y se ha visto su utilidad como herramienta docente. Los patrones podrían formar parte durante todo el proceso de desarrollo hipermedia. Podrían comenzar su participación como herramienta comunicativa durante la captura y especificación de requisitos en la fase de Análisis, lo que permitiría verificar el cumplimiento de un requisito y dejar constancia de cómo se ha resuelto un problema. Desde el punto de vista de la fase de Evaluación, los patrones podrían ser utilizados para evaluar la potencial usabilidad del sistema y detectar así las carencias tanto en las fase de análisis como de diseño.

- Estudiar el potencial del modelo de representación semántico para la formalización de otros patrones específicos del dominio, como pueda ser el de interacción persona-ordenador.

  La arquitectura definida en el modelo de representación se basa en la separación del dominio de los componentes del patrón del dominio de las aplicaciones hipermedia. Por lo tanto, este último dominio puede ser intercambiado por otro dominio de aplicación y permitir así representar cualquier patrón de diseño específico del dominio. Un dominio interesante es el de los patrones de interacción persona-ordenador, en el que se puede encontrar ya una especificación basada en XML [5].

- Estudiar la inclusión de una fase de evaluación en el proceso de elaboración, aplicación y mantenimiento de los patrones de diseño.

Aunque los patrones son fruto de la experiencia y por ese motivo ya han demostrado su utilidad en sí mismos, sin embargo su transcripción al lenguaje natural o el desarrollo de lenguajes de patrones no queda exento de mejoras y de carencias. Además, puesto que los lenguajes de patrones tienden a cambiar con el tiempo, es decir, nuevos patrones pueden ser incluidos o el uso de los mismos puede cambiar, es necesario detectar y eliminar las inconsistencias. Por ello, es necesario incluir una fase de evaluación en el cual tanto diseñadores noveles como expertos tomen el pulso de la utilidad y usabilidad de los patrones, y a partir de la cual poder mejorar aspectos como su formato, narrativa o enfoque, e incluso poder detectar si los patrones son obsoletos.

Referencias bibliográficas

[Adams *et al.*, 2001] J. Adams, S. Koushik, G. Vasudeva, y G. Galambos. *Patterns for e-business: A Strategy for Reuse*. IBM, 2001.

[Aedo *et al.*, 2002a] I. Aedo, P. Díaz, C. Fernández, y J. de Castro. Supporting efficient multinational disaster response through a web-based system. In *Electronic Government: First International Conference, EGOV 2002*, 2002.

[Aedo *et al.*, 2002b] I. Aedo, S. Montero, y P. Díaz. Supporting personalization in a web based course through the definition of role-based access policies. *Interactive Educational Multimedia*, 4:40–52, 2002.

[Aedo *et al.*, 2003] I. Aedo, P. Díaz, y S. Montero. A methodological approach for hypermedia security modeling. *Information & Software Technology*, 45(5):229–239, 2003.

[Albin-Amiot y Guéhéneuc, 2001] H. Albin-Amiot y Y. Guéhéneuc. Meta-modeling design patterns: Application to pattern detection and code synthesis. In *Proceedings of the ECOOP Workshop on Automating Object-Oriented Software Development Methods*, pages 57–64, June 2001.

[Alexander *et al.*, 1977] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, y S. Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, 1977.

[Alexander *et al.*, 1985] C. Alexander, H. Davis, J. Martinez, y D. Corner. *The Production of Houses*. Oxford University Press, NY, 1985.

[Alexander, 1979] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[Alur *et al.*, 2003] D. Alur, D. Malks, y J. Crupi. *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall PTR; 2nd edition, 2003.

[Ambler, 1998] S.W. Ambler. *Process Patterns : Building Large-Scale Systems Using Object Technology*. Cambridge University Press, 1998.

[Amitay *et al.*, 2003] E. Amitay, D. Carmel, A. Darlow, R. Lempel, y A. Soffer. The connectivity sonar: detecting site functionality by structural patterns. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 38–47. ACM Press, 2003.

[Antoy y Hanus, 2002] S. Antoy y M. Hanus. Functional logic design patterns. In *Proc. of the 6th International Symposium on Functional and Logic Programming (FLOPS 2002)*, 2002.

[Aridor y Lange, 2003] Y. Aridor y D. B. Lange. A pattern language for motivating the use of agents. In *Proceedings of Agent-Oriented Information Systems, 5th International Bi-Conference Workshop, AOIS 2003*, pages 142–157, 2003.

[Baresi *et al.*, 2002] L. Baresi, S. Morasca, y P. Paolini. An empirical study on the design effort of web applications. In *Proceedings of 3rd International Conference on Web Information Systems Engineering (WISE'02)*. IEEE Computer Society, 2002.

[Barry y Lang, 2001] C. Barry y M. Lang. A survey of multimedia and web development techniques and methodology usage. *IEEE Multimedia*, pages 52–60, 2001.

[Berners-Lee *et al.*, 2001] T. Berners-Lee, J. Hendler, y O. Lassila. The semantic web. *Scientific American*, May 2001.

[Bernstein, 1998] M. Bernstein. Patterns of hypertext. In *Proceedings of the 9th ACM Conference on Hypertext, Hypermedia Application Design*, pages 21–29, Pittsburgh, PA, 1998.

[Blakley y Heath, 2004] B. Blakley y C. Heath. Security design patterns. Technical report, The Open Group, 2004.

[Bolchini *et al.*, 2003] D. Bolchini, P. Paolini, y G. Randazzo. Adding hypermedia requirements to goal-driven analysis. In *11th IEEE International Requirements Engineering Conference (RE'03)*, pages 127–137, 2003.

[Bolchini, 2000] D. Bolchini. Web design patterns: Improving quality and performance in web application design. Master's thesis, Universit della Svizzera Italiana, 2000.

[Bonura *et al.*, 2002] D. Bonura, R. Culmone, y E. Merelli. Patterns for web applications. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 739–746. ACM Press, 2002.

[Borchers, 2000] J. O. Borchers. A pattern approach to interaction design. In *Proceedings of the conference on Designing interactive systems*, pages 369–378. ACM Press, 2000.

[Borchers, 2001] J. O. Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, 2001.

[Borchers, 2002] J. Borchers. Teaching hci design patterns: Experience from two university courses. In *CHI 2002 International Conference on Human Factors and Computing Systems*, 2002.

[Borst *et al.*, 1997] W.Ñ. Borst, J. M. Akkermans, y J. L. Top. Engineering ontologies. *International Journal of Human-Computer Studies*, 46:365–406, 1997.

[Buschmann *et al.*, 1996] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, y M. Stal. *A System of Patterns: Pattern-Oriented Software Architecture*. Wiley, 1996.

[Buschmann *et al.*, 2001] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, y M. Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley, 2001.

[Bush, 1945] V. Bush. As we may think. *Atlantic Monthly*, 176(1):101–108, 1945.

[Ceri *et al.*, 2000] S. Ceri, P. Fraternali, y A. Bongio. Web modeling language (WebML): a modeling language for designing web sites. *WWW9 / Computer Networks*, 33(1-6):137–157, 2000.

[Chambers *et al.*, 2000] C. Chambers, B. Harrison, y J. Vlissides. A debate on language and tool support for design patterns. In *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 277–289. ACM Press, 2000.

[Coplien, 1998] J. O. Coplien. *Software Design Patterns: Common Questions and Answers*, pages 311–320. Cambridge University Press, New York, 1998.

[Cornils y Hedin, 2000] A. Cornils y G. Hedin. Tool support for design patterns based on reference attribute grammars. In *Proc. of WAGA'00, Ponte de Lima, Portugal*, 2000.

[Cybulski y Linden, 1999] J.L. Cybulski y T. Linden. *Pattern Languages of Program Design*, volume 4, chapter Composing Multimedia Artefacts for Reuse, pages 461–488. Addison-Wesley Longman, 1999.

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

[Díaz *et al.*, 1999] P. Díaz, I. Aedo, y F. Panetsos. 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)*, pages 213–228, Paris, France, September 1999.

[Díaz *et al.*, 2001a] P. Díaz, I. Aedo, y S. Montero. Ariadne, a development method for hypermedia. In *proceedings of Dexa 2001*, volume 2113 of *LNCS*, pages 764–774, 2001.

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

[Díaz *et al.*, To be published] P. Díaz, S. Montero, y I. Aedo. Modelling hypermedia and web applications: the ariadne development method. *Information System*, To be published.

[De Troyer, 2001] O. De Troyer. *Audience-driven web design*. IDEA Group Publishing, 2001.

[Dearden *et al.*, 2002] A. Dearden, J. Finlay, E. Allgar, y B. McManus. Using pattern languages in participatory design. In *Proceedings of the Participatory Design Conference*, pages 104 – 113, 2002.

[Deshpande y Hansen, 1998] Y. Deshpande y S. Hansen. Web engineering:creating a discipline among disciplines. *IEEE Multimedia*, 2(8):82–87, 1998.

[Devedzic, 2002] V. Devedzic. Understanding ontological engineering. *Commun. ACM*, 45(4):136–144, 2002.

[Dong, 2003] J. Dong. Representing the applications and compositions of design pattern in uml. In *the Proceedings of the Eighteenth Annual ACM Symposium on Applied Computing (SAC)*, pages 1092–1098, 2003.

[Ebersole, 1997] S. Ebersole. Cognitive issues in the design and deployment of interactive hypermedia: Implications for authoring www sites. *Interpersonal Computing and Technology*, 1997.

[Eden *et al.*, 1997] A. H. Eden, A. Yehudai, y J. Gil. Precise specification and automatic application of design patterns. In *Proc. of International Conference on Automated Software Engineering (ASE '97)*, pages 143–152, Lake Tahoe, CA, USA, 1997.

[Erickson, 2000] T. Erickson. Lingua francas for design: sacred places and pattern languages. In *Proceedings of the conference on Designing interactive systems*, pages 357–368. ACM Press, 2000.

[Escalona y Koch, 2003] M. J. Escalona y N. Koch. Requeriments engineering for web applications: A comparative study. *Journal of Web Engineering*, 2(3):192–212, 2003.

[Euzenat, 2002] J. Euzenat. Eight questions about semantic web annotations. *IEEE Intelligent Systems*, 2002.

[Fensel *et al.*, 2000] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, y M. Klein. Oil in a nutshell. In *Knowledge Acquisition, Modeling, and Management, EKAW-2000*, 2000.

[Fernández *et al.*, 1996] E. B. Fernández, M.M. Krishnakumar, R.N.and Larrondo-Petrie, y Y. Xu. High-level security issues in multimedia/hypertext systems. *Communications and Multimedia Security II*, pages 13–24, 1996.

[Fernández y Pan, 2001] E.B. Fernández y R. Pan. A pattern language for security models. In *Procs. of PloP 2001*, 2001.

[Florijn *et al.*, 1997] G. Florijn, M. Meijers, y P. van Winsen. Tool support for object-oriented patterns. In *Proceedings of ECOOP'97*, Finland, 1997.

[Fowler, 1997a] M. Fowler. *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1997.

[Fowler, 1997b] M. Fowler. Dealing with roles. In *The PLoP '97*, 1997.

[France *et al.*, 2004] R. B. France, D. Kim, S. Ghosh, y E. Song. A uml-based pattern specification technique. *IEEE Trans. Softw. Eng.*, 30(3):193–206, 2004.

[Fraternali, 1999] P. Fraternali. Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys*, 31(3):227–263, 1999.

[Gaedke y Rehse, 2000] M. Gaedke y J. Rehse. Supporting compositional reuse in component-based web engineering. In *Proceedings of the 2000 ACM symposium on Applied computing*, pages 927–933. ACM Press, 2000.

[Gamma *et al.*, 1994] E. Gamma, R. Helm, R. Johnson, y J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[Garrido *et al.*, 1997] A. Garrido, G. Rossi, y D. Schwabe. Patterns systems for hypermedia. In *Proceedings of The 3rd Pattern Language of Programming Conference*, 1997.

[Garzotto *et al.*, 1993] F. Garzotto, P. Paolini, y D. Schwbe. HDM- a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, 11(1):1–26, 1993.

[Garzotto *et al.*, 1997] F. Garzotto, L. Mainetti, y P. Paolini. Designing model hypermedia applications. In *Hypertext 97, The Eigth ACM Conference on Hypertext, University of Southampton, UK, April 6-11, 1997*, pages 38–47. ACM, 1997.

[Garzotto *et al.*, 1999] F. Garzotto, P. Paolini, D. Bolchini, y S. Valenti. 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*, pages 293–306, 1999.

[German y Cowan, 1999] D. German y D. Cowan. Three hypermedia design patterns. In *Proceedings of the HT99 Workshop on Hypermedia Development Design Patterns in Hypermedia*, 1999.

[German y Cowan, 2000] D. German y D. Cowan. Towards a unified catalog of hypermedia design patterns. In *Proceedings of 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, 2000.

[Gómez *et al.*, 2001] J. Gómez, C. Cachero, y O. Pastor. Conceptual modeling of device-independent web applications. *IEEE MultiMedia*, 8(2):26–39, 2001.

[Gómez-Pérez *et al.*, 2004] A. Gómez-Pérez, O. Corcho, y M. Fernández-López. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004.

[Gomes *et al.*, 2002] P. Gomes, F. Pereira, P. Paiva, N. Seco, P. Carreiro, J.L. Ferreira, y C. Bento. Using CBR for automation of software design pattern. In *Proceedings of the European Conference Case-Based Reasoning (ECCBR'02)*, pages 534–548, 2002.

[Gould *et al.*, 1997] J. D. Gould, S. J. Boies, y J. Ukelson. *Handbook of human-computer interaction*, chapter How to design usable systems, pages 231–254. Elsevier Science, 1997.

[Graham, 2003a] I. Graham. *A Pattern Language for Web Usability*. Addison Wesley Professional, 2003.

[Graham, 2003b] I. Graham. *A Pattern Language for Web Usability*. Addison-Wesley, 2003.

[Greene *et al.*, 2003] S. L. Greene, P. M. Matchen, L. Jones, John C. Thomas, y Matt Callery. Tool-based decision support for pattern assisted development. In *Perspectives on HCI Patterns: Concepts and Tools Workshop at CHI 2003*, 2003.

[Gross y Yu, 2000] D. Gross y E. Yu. From non-functional requirements to design through patterns. In *Proceedings of the 6th International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ 2000)*, 2000.

[Gruber, 1993] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Guarino, 1997] N. Guarino. Understanding, building, and using ontologies: .ª commentary to using explicit ontologies in kbs development ", by van heijst, schreiber, and wielinga. *International Journal of Human and Computer Studies*, 46:293–310, 1997.

[Guerrero y Fuller, 2001] L. A. Guerrero y D. A. Fuller. A pattern system for the development of collaborative applications. *Information & Software Technology*, 43(7):457–467, 2001.

[Halasz y Schwartz, 1990] F. G. Halasz y M. Schwartz. The dexter hypertext reference model. In *Proc. of World Conference of Hypertext*, pages 95–133, 1990.

[Hardman *et al.*, 1994] L. Hardman, D.C.A. Bulterman, y G. Van Rossum. The amsterdam hypermedia model. *Communications of the ACM*, 37:50–62, 1994.

[Hendler y McGuinness, 2000] J. Hendler y D. L. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent System*, 6(15):25–29, 2000.

[Hennicker y Koch, 2000] R. Hennicker y N. Koch. A UML-based methodology for hypermedia design. In *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings*, LNCS, pages 410–424, 2000.

[Henninger *et al.*, 2003] S. Henninger, M. Keshk, y R. Kinworthy. Capturing and disseminating usability patterns with semantic web technology. In *CHI Workshop on Perspectives on HCI Patterns: Concepts and Tools*, 2003.

[Horrocks y Patel-Schneider, 2003] I. Horrocks y P. F. Patel-Schneider. Reducing owl entailment to description logic satisfability. In *The Semantic Web - ISWC 2003*, pages 17–29, 2003.

[Hubscher y Frizell, 2002] R. Hubscher y S. Frizell. Supporting the application of design patterns in web-course design. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 544–549, 2002.

[IEE, 1990] IEEE. *IEEE Std 610.12 Standard Glossary of Software Engineering Terminology*, 1990.

[Isakowitz *et al.*, 1998] T. Isakowitz, A. Kamis, y M. Koufaris. The extended RMM methodology for web publishing. Technical Report Working Paper IS98 -18, Center for Research on Information Systems, 1998.

[Kappel *et al.*, 2000] G Kappel, W. Retschitzegger, y W. Schwinger. Modeling customizable web applications - a requirement's perspective. In *Kyoto International Conference on Digital Libraries*, 2000.

[Kardell, 1997] Magnus Kardell. A classification of object-oriented design patterns. Master's thesis, Umeå University, 1997.

[Keidel, 1995] Robert W. Keidel. *Seeing Organizational Patterns: A New Theory and Language of Organizational Design*. Berrett-Koehler Publishers, 1995.

[Koch y Rossi, 2000] N. Koch y G. Rossi. Patterns for adaptive web applications. In *EuroPlop*, 2000.

[Konrad *et al.*, 2003] S. Konrad, B. H. C. Cheng, L. A. Campbell, y R. Wassermann. Using security patterns to model and analyze security requirements. In *Proc. of the Requirements for High Assurance Systems Workshop (RHAS03) as part of the IEEE Joint International Conference on Requirements Engineering (RE03)*, Monterey Bay, CA, USA, September 2003.

[Landay y Borriello, 2003] J. A. Landay y G. Borriello. Design patterns for ubiquitous computing. *Computer*, 36(8):93–95, 2003.

[Lenaerts y Manderick, 1998] T. Lenaerts y B. Manderick. Building a genetic programming framework: The added-value of design patterns. In *Proceedings of EuroGP'98*, pages 196–20, 1998.

[Lowe y Hall, 1999] D. Lowe y W. Hall. *Hypermedia and the Web: an engineering approach*. Chichester: John Wiley and Sons, 1999.

[Lucrédio *et al.*, 2003] D. Lucrédio, A. Alvaro, E. Santana de Almeida, y A. F. do Prado. Mvcase tool- working with design patterns. In *SugarloafPLoP 2003 Conference*, 2003.

[Lyardet *et al.*, 1998] F. Lyardet, G. Rossi, y D. Schwabe. Patterns for dynamic websites. In *Proceedings of The 4th Pattern Languages of Programming Conference*, 1998.

[Lyardet *et al.*, 1999] F. Lyardet, G. Rossi, y D. Schwabe. Discovering and using design patterns in the www. *Multimedia Tools and Applications*, 8:293–308, 1999.

[Mcguinness *et al.*, 2002] D.L. Mcguinness, R. Fikes, J. Hendler, y L.A. Stein. DAML+OIL: an ontology language for the Semantic Web . *IEEE Intelligent System*, 5(17):72–80, 2002.

[Molina, 2003] P. J. Molina. *Especificación de interfaz de usuario: de los requisitos a la espeificación automática*. PhD thesis, Universidad Politécnica de Valencia, 2003.

[Montero *et al.*, 2002] S. Montero, P. Díaz, y I. Aedo. Requirements for hypermedia development methods: A survey of outstanding methods. In *Proc. of Advanced Information Systems Engineering, 14th International Conference, CAiSE*, volume 2348 of *LNCS*, pages 747–751, 2002.

[Montero *et al.*, 2003a] S. Montero, P. Díaz, y I. Aedo. Formalization of web design patterns using ontologies. In *Proc. of First International Atlantic Web Intelligence Conference, AWIC*, volume 2663 of *LNCS*, pages 179–188. Springer, 2003.

[Montero *et al.*, 2003b] S. Montero, P. Díaz, y I. Aedo. A framework for the analysis and comparison of hypermedia design methods. In *Proc. of The IASTED International Conference on Software Engineering (SE'2003)*, pages 1053–1058, 2003.

[Montero *et al.*, 2003c] S. Montero, P. Díaz, y I. Aedo. Toward hypermedia design methods for the semantic web. In *14th International Workshop on Database and Expert Systems Applications (DEXA'03)*, pages 762–767. IEEE Computer Society, 2003.

[Montero *et al.*, 2004a] S. Montero, P. Díaz, y I. Aedo. Ariadnetool: A design toolkit for hypermedia applications. *Journal of Digital Information*, 5(2), 2004.

[Montero *et al.*, 2004b] S. Montero, P. Díaz, y I. Aedo. A semantic representation for domain-specific patterns. In *Metainformatics International Symposium, MIS 2004*, volume 3511 of *LNCS*. Springer, 2004.

[Montero *et al.*, 2005] S. Montero, P. Díaz, y I. Aedo. *Ingeniería de la Web y Patrones de Diseño*, chapter 12 Integración de patrones en el proceso de diseño de aplicaciones hipermedia. 2005.

[Nanard *et al.*, 1998] M. Nanard, J. Nanard, y P. Kahn. Pushing reuse in hypermedia design: golden rules, design patterns and constructive templates. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space—structure in hypermedia systems*, pages 11–20. ACM Press, 1998.

[Nanard y Nanard, 1995] J. Nanard y M. Nanard. Hypertext design environments and the hypertext design process. *Comm. of the ACM*, 38(8):49–56, 1995.

[Nanard y Nanard, 1999] J. Nanard y M. Nanard. Toward an hypermedia design patterns space. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[Neches *et al.*, 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, y W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.

[Nielsen, 1995] J. Nielsen. *Hypertext and Hypermedia: the Internet and Beyond*. Academic Press, 1995.

[Nielsen, 2000] J. Nielsen. *Designing Web Usability*. New Riders Publishing, 2000.

[Noack *et al.*, 2000] J. Noack, H. Mehmaneche, H. Mehmaneche, y A. Zendler. Architectural patterns for web applications. In *18th IASTED Conference*, 2000.

[Noble, 1998] J. Noble. Towards a pattern language for object oriented design. In *Proocedings of Technology of Object-Oriented Languages*, pages 2–13, 1998.

[Paolini y Garzotto, 1999] P. Paolini y F. Garzotto. Design patterns for www hypermedia: problems and proposals. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia. Hypertext'99*, 1999.

[Peters, 2003] J. F. Peters. Design patterns in intelligent systems. In *Foundations of Intelligent Systems, 14th International Symposium, ISMIS 2003*, pages 262–269, 2003.

[Porter y Calder, 2004] R. Porter y P. Calder. Patterns in learning to program: an experiment? In *Proceedings of the sixth conference on Australian computing education*, pages 241–246. Australian Computer Society, Inc., 2004.

[Poulin, 1996] Jeffrey S. Poulin. *Measuring software reuse: principles, practices, and economic models*. Addison-Wesley Longman Publishing Co., Inc., 1996.

[Preece *et al.*, 2002] J. Preece, Y. Rogers, y H. Sharp. *Interaction Design: Beyond Human-Computer Interaction*. New York, NY: John Wiley & Sons, 2002.

[Riehle y Zullighoven's, 1996] D. Riehle y H. Zullighoven's. Understanding and using patterns in software development. *Theory and Practice of Object Systems*, 2(1), 1996.

[Rising, 2003] L. Rising. Pattern forms. In *Proceedings of Viking PLOP 2003*, 2003.

[Rossi *et al.*, 1996] G. Rossi, A. Garrido, y S. Carvalho. *Design Patterns for Object-Oriented Hypermedia Applications. Pattern Languages of Programs II*. Addison-Wesley, 1996.

[Rossi *et al.*, 1997] G. Rossi, D. Schwabe, y A. Garrido. Design reuse in hypermedia application development. In *Proceedings of the 8th. ACM Conference on Hypertext: Hypertext'97*, Southampton, England, 1997.

[Rossi *et al.*, 1999] G. Rossi, F. Lyardet, y D. Schwabe. Developing hypermedia applications with methods and patterns. *ACM Computing Surveys*, 31(4es):8, 1999.

[Rossi *et al.*, 2000a] G. Rossi, D. Schwabe, y F. Lyardet. User interface patterns for hypermedia application. In *Proceedings of Advanced Visual Interfaces 2000*, pages 136–142, 2000.

[Rossi *et al.*, 2000b] G. Rossi, D. Schwabe, y F. Lyardet. User interface patterns for hypermedia applications. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 136–142. ACM Press, 2000.

[Rossi *et al.*, 2001a] G. Rossi, D. Schwabe, J. Danculovic, y L. Miaton. Patterns for personalized web applications. In *Proceedings of EuroPlop´01,*, pages 423–436, 2001.

[Rossi *et al.*, 2001b] G. Rossi, D. Schwabe, y M. Guimaraes. Designing personalized web applications. In *Proceedings of the World Wide Web Conference (WWW'10). Hong Kong*, pages 275–284, May 2001.

[Rossi y Lyardet, 1999] D. Rossi, G.and Schwabe y F. Lyardet. Integrating patterns into the hypermedia development process. *New Review of Hypermedia and Multimedia*, 5:59–80, 1999.

[Rumbaugh *et al.*, 1998] J. Rumbaugh, J. Jacobson, y G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1998.

[Salingaros, 2000] N. A. Salingaros. The structure of pattern languages. *Architectural Research Quarterly*, 4:149–161, 2000.

[Schumacher, 2003] M. Schumacher. *Security Engineering With Patterns: Origins, Theoretical Models, and New Applications*. Springer-Verlag, 2003.

[Schwabe y Rossi, 1998] D. Schwabe y G. Rossi. An object oriented approach to web-based application design. *Theory and practice of object systems*, 4(4):207–225, 1998.

[Seffah, 2003] A. Seffah. Learning the ropes: human-centered design skills and patterns for software engineers' education. *Interactions*, 10(5):36–45, 2003.

[Sicilia *et al.*, 2003] M. A. Sicilia, E. García, I. Aedo, y P. Díaz. A literature-based approach to annotation and browsing of web resources. *Information Research*, 8(2), 2003.

[Sierra, 1986] R. Sierra. *Tesis doctorarles y trabajos de investigación cientſfica*. Thomson editores Spain Paraninfo, S.A., 1986.

[Studer *et al.*, 1998] R. Studer, V. R. Benjamins, y D. Fensel. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1-2):161–197, 1998.

[Sunyé *et al.*, 2000] G. Sunyé, A. Le Guennec, y J.M. Jézéquel. Design patterns application in uml. In *Proceedings of ECOOP*, pages 44–62, 2000.

[Tahvildari y Kontogiannis, 2002] A. Tahvildari y K. Kontogiannis. On the role of design patterns in quality-driven re-engineering. In *Sixth European Conference on Software Maintenance and Reengineering*, pages 230–240, 2002.

[Taibi y Ngo, 2003] T. Taibi y D. C. Ling Ngo. Formal specification of design patterns - a balanced approach. *Journal of Object Technology*, 2003.

[Todd *et al.*, 2004] E. Todd, E. Kemp, y C. Phillips. What makes a good user interface pattern language? In Andy Cockburn, editor, *Fifth Australasian User Interface Conference (AUIC2004)*, volume 28 of *CRPIT*, pages 91–100, Dunedin, New Zealand, 2004. ACS.

[van Duyne *et al.*, 2002] D. K. van Duyne, J. A. Landay, y J. I. Hong. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Addison-Wesley, 2002.

[Weiss, 2003] M. Weiss. Patterns for web applications. In *Pattern Languages of Programming (PLoP)*, 2003.

[Zimmer, 1995] W. Zimmer. Relationships between design patterns. In *Pattern Languages of Program Design*, pages 345–364. Addison-Wesley, 1995.

Referencias electrónicas

[1] http://www.webratio.com/.

[2] http://www.visualwade.com/.

[3] Annotation tools. http://annotation.semanticweb.org/tools.

[4] Ifip tc13 hci patterns task group. http://hcipatterns.org/tiki-index.php.

[5] The pattern language markup language (plml). http://media.informatik.rwth-aachen.de/patterns/tiki/tiki-download_file.php?fileId=7.

[6] the security patterns community. http://securitypatterns.org/.

[7] Web services. http://www.w3.org/2002/ws/.

[8] *2nd Workshop on Hypermedia Development: Design Patterns in Hypermedia, in Conjunction with ACM Conference Hypertext'99*, 1999.

[9] Access control and authorization. a guide to building secure web applications. http://www.cgisecurity.com/owasp/html/ch08.html.

[10] http://arce.proteccioncivil.org/.

[11] Adm- ariadne development method. http://ariadne.dei.inf.uc3m.es.

[12] Web pattern repository. http://www.designpattern.lu.unisi.ch/PatternsRepository.htm.

[13] A pattern language for living communication. http://cpsr.org/program/sphere/patterns/.

[14] Patterns for concurrent, parallel, and distributed systems. http://www.cs.wustl.edu/ schmidt/patterns-ace.html.

[15] The patterns of a conservation economy. http://www.conservationeconomy.net/.

[16] ezowl. http://iweb.etri.re.kr/ezowl/index.html.

[17] I. Graham. A pattern language for web usability. http://wupatterns.com/.

[18] V. Haarslev, R. Möller, and M. Wessel. Racer system. http://www.sts.tu-harburg.de/ r.f.moeller/racer/download.html.

[19] Hillside group. http://www.hillside.net/, 2004.

[20] Ibm ease of use. http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/572.

[21] M. L. Irons. Patterns for personal web sites. http://www.rdrop.com/ half/Creations/Writings/Web.patterns/index.html, 2004.

[22] http://jena.sourceforge.net/.

[23] Technical Comitee T2 National Comitee for Information Technology standards. http://logic.stanford.edu/kif/dpans.html.

[24] http://www.w3.org/TR/owl-features/.

[25] The pedagogical patterns project. http://www.pedagogicalpatterns.org/.

[26] Pattern languages of programs conferences. http://hillside.net/conferences/, 2004.

[27] Protégé owl plugin. http://protege.stanford.edu/plugins/owl/.

[28] Resource description framework (rdf). http://www.w3c.org/RDF.

[29] http://jmis.bentley.edu/rmm/.

[30] http://www.w3.org/2001/sw/.

[31] Siteview. http://www.d-sciencelab.com/siteview/.

[32] J. Tidwell. Common ground: A pattern language for human-computer interface design. http://www.mit.edu/ jtidwell/common_ground.html.

[33] M. van Welie. Web pattern repository. http://www.welie.com/index.html.

Anexo A

Un lenguaje de patrones hipermedia

En este anexo se muestra de manera completa los patrones que conforman el lenguaje presentado en la sección 4.3. En dicha sección se recogía el objetivo global del lenguaje, un índice comentado de los patrones y un mapa que mostraba la posición de cada patrón dentro del lenguaje, organizados de forma jerárquica de mayor a menor nivel de abstracción en un proceso de refinamiento, y las relaciones semánticas de uso existente entre ellos.

Antes de proceder a la descripción de cada patrón, es necesario señalar que debido a las diferentes fuentes bibliográficas utilizadas, ha sido necesario por un lado, reescribir dichos patrones para que presenten un formato único y por otro, rehacer parte de sus contenidos con la intención de que cada patrón sea un paso más en la consecución final del objetivo del lenguaje, formando así una red asociativa de patrones.

A continuación se presenta el formato que seguirán los patrones, basado en el formato presentado en *"A pattern language"* [Alexander *et al.*, 1977]:

- El `[ID]Nombre` del patrón en tipografía de `máquina de escribir`.
- Un breve párrafo orientado al **contexto** de este patrón con respecto a los anteriores, y una situación en el cual debería ser considerado.
- Una exposición breve del **problema**.
- Una **discusión** del problema, especialmente centrado en por qué este problema es difícil y no se resuelve trivialmente. La discusión explora las motivaciones del problema y dirige a la resolución de dichas motivaciones.

- Una **solución** concisa al problema.
- Un **diagrama** que esquematiza la solución,
- Un párrafo concluyente que muestre como ese patrón está **relacionado con otros patrones** del lenguaje, que a continuación deberían ser consultados, junto con un razonamiento para cada elección.

La intención global del lenguaje de patrones y las motivaciones que han llevado a la presente recopilación se presentan en forma del patrón `[A]Hypermedia Applications`, que a su vez servirá como punto de partida para dicho lenguaje.

El resto de patrones están divididos conceptualmente por grupos de acuerdo con el aspecto de diseño con el que trata el problema que presentan. Cada grupo recibe el nombre del patrón de más alto nivel que se encuentra en dicho grupo, resumiendo así la intención del conjunto.

A.1. [A]Hypermedia Application

`[A]Hypermedia Application`

Contexto Si la aplicación a 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 [Díaz *et al.*, 1999]. 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 [Nanard y Nanard, 1995]. 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. [Díaz *et al.*, 1999]. 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 [Kappel *et al.*, 2000]. La personalización 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 [Aedo *et al.*, 2003]. 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 independien-

temente 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.

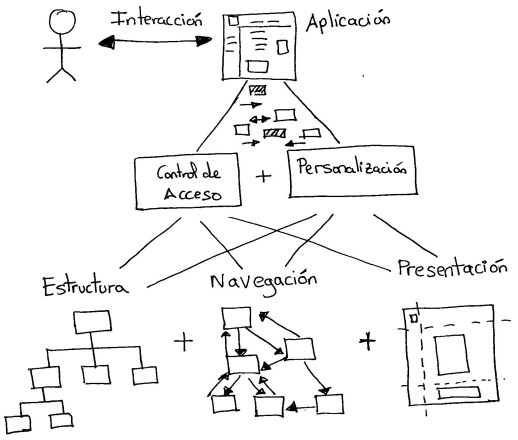


Figura A.1: Separar la estructura de la información, de la navegación y de la presentación, hace que la aplicación sea accesible a un mayor número de personas y dispositivos

Patrones relacionados El patrón `[AE1]User-centered Structure` ayuda a organizar la información teniendo en cuenta las necesidades de los usuarios. El patrón `[AN1]Multiple Ways to Navigate` 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]Aesthetics` 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 interacción entre el usuario y la aplicación mediante `[AI1]Interaction`. Si se desea que la aplicación sea utilizada por el mayor número de usuarios, se puede seguir el patrón

[AZ1]Personalization, a la vez que es necesario controlar quién puede hacer qué cosas en la aplicación, mediante el patrón [AS1]Access Control.

Fuentes

Formalización

```
<rdf:RDF
    xmlns="http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl/A#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:domain="http://hypatterns.no-ip.info/ontologies/hypermedia.owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:pattern="http://hypatterns.no-ip.info/ontologies/pattern.owl#"
    xmlns:hdp="http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl#">
  <pattern:Actor rdf:ID="Actor"/>
  <domain:Hyperdocument rdf:about="#aplicación hipermedia"/>
  <hdp:HypermediaPattern rdf:ID="A">
    <pattern:discussion>No todos los usuarios poseen las mismas motivaciones</pattern:discussion>
    <pattern:context>información de manera personalizada</pattern:context>
    <pattern:context>comportamientos interactivos</pattern:context>
    <pattern:context>contenidos multimedia</pattern:context>
    <pattern:problem>útil y fácil de usar</pattern:problem>
    <pattern:problem rdf:resource="#aplicación hipermedia"/>
    <pattern:discussion>Cada dominio de aplicación se caracteriza por una serie de
     conceptos y de relaciones</pattern:discussion>
    <pattern:discussion>necesidad de preservar la seguridad de la
     información</pattern:discussion>
    <hdp:level>High</hdp:level>
    <pattern:name>Hypermedia Application</pattern:name>
    <pattern:discussion>se está potenciando la accesibilidad mediante la personalización,
     tanto de la información como de las funcionalidades</pattern:discussion>
    <pattern:context>manipulación de la información</pattern:context>
    <pattern:discussion>necesidad de organizar y armonizar dicha información en diferentes
     dimensiones como el tiempo y el espacio </pattern:discussion>
    <pattern:discussion>el usuario corre el riesgo de perderse en el hiperespacio,
     alcanzando una posición de la cual es incapaz de salir hacia un punto
      conocido</pattern:discussion>
    <pattern:context>seguridad de la información</pattern:context>
    <pattern:diagram>hypermediaaplicationpattern.jpg</pattern:diagram>
    <hdp:usesTo rdf:resource="AN1#AN1"/>
    <hdp:usesTo rdf:resource="AE1#AE1"/>
    <pattern:context>facilitar el acceso</pattern:context>
    <hdp:usesTo rdf:resource="AS1#AS1"/>
    <hdp:usesTo rdf:resource="AI1#AI1"/>
    <hdp:usesTo rdf:resource="AZ1#AZ1"/>
    <hdp:usesTo rdf:resource="AP1#AP1"/>
  </hdp:HypermediaPattern>
</rdf:RDF>
```

A.1.1. [AE1]User-centered Structure

Aunque la navegación es la característica principal de las aplicaciones hipermedia, existe un espacio de información subyacente que debe ser organizado para ayudar a que el

acceso a la misma se realice de manera más intuitiva y sencilla. Los patrones aquí recogidos bajo el nombre de `User-centered Structure` abordan problemas relacionados con cómo acometer la organización de la información y de las tareas contenidas en la aplicación, independientemente de su dominio y teniendo como fundamento cumplir las expectativas de los usuarios.

`[AE1]User-centered Structure`

Contexto Se está construyendo una `[A]Hypermedia Application` 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?

Discusión 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 expectativas de cómo encontrar los diferentes tipos de información y cómo llevar a cabo ciertas tareas en la aplicación, por ejemplo, cuando un usuario accede a una aplicación que ofrece servicios, espera que se le faciliten los servicios o la 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 encaje la organización de la información 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 tareas como les gustaría a los usuarios realizarlas.

Relacionado con Para organizar la información de manera jerárquica, utilizar el patrón `[ME1]Hierarchical Organization`. Cuando exista una secuencia en la información para llevar a cabo tareas se debería utilizar el patrón `[ME2]Task-Based`

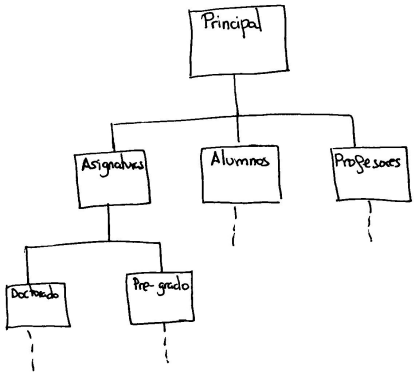


Figura A.2: Balancear entre anchura y profundidad; si una jerarquía es demasiado ancha, los usuarios pueden tener que elegir entre demasiadas opciones; si es demasiado profunda, van a tener que navegar demasiado para alcanzar un determinado nodo

`Organization`. Cada subcategoría de la estructura o tarea debería ser presentada al usuario con la ayuda del patrón `[BE1]Collection Center`. Además, para definir lo que sería cada una de las unidades de información de la estructura, se puede consultar el patrón `[BE2]Node as a Single Unit`. Para finalizar se debería definir un raíz de la estructura organizativa mediante el patrón `[BE3]Home Page`.

Independientemente de la estructura de la información resultante, es necesario proporcionar a los usuarios diferentes modos de navegación, véase el patrón `[AN1]Multiple Ways to Navigate`. La información en combinación con las herramientas de navegación deben ser presentada al usuario, véase el patrón `[AP1]Aesthetics`. El usuario debe tener conocimiento del estado de la interacción con la aplicación, véase el patrón `[AI1]Interaction`. Además, puede ser que se quiera personalizar la información en función de los usuarios de la aplicación, véase el patrón `[AZ1]Personalization`, y/o controlar su acceso, véase el patrón `[AS1]Control Access`.

Fuentes [20], [17]. Para profundizar en características más concretas relacionadas con el dominio de la aplicación se pueden aplicar los patrones recogidos en [van Duyne *et al.*, 2002] como `Grupo A. Site Genres`.

Formalización

```
<domain:Hyperdocument rdf:ID="aplicación"/>
 <domain:CompositeComponent rdf:ID="información"/>
```

```
<hdp:HypermediaPattern rdf:ID="AE1">
  <pattern:solution rdf:resource="#aplicación"/>
  <pattern:diagram>usercenteredstructurepattern.jpg</pattern:diagram>
  <pattern:problem>estructurar la información y las tareas</pattern:problem>
  <hdp:level>High</hdp:level>
  <hdp:specializedBy rdf:resource="ME1#ME1"/>
  <pattern:discussion>existe un espacio de información subyacente que debe ser
   organizado</pattern:discussion>
  <hdp:usesTo rdf:resource="BE3#BE3"/>
  <pattern:problem>acceso intuitivo y sencillo</pattern:problem>
  <pattern:context rdf:resource="A#A"/>
  <pattern:discussion> las personas desarrollan ciertas expectativas de cómo encontrar
   los diferentes tipos de información y cómo llevar a cabo ciertas
    tareas</pattern:discussion>
  <pattern:problem rdf:resource="#aplicación"/>
  <pattern:solution rdf:resource="#información"/>
  <hdp:specializedBy rdf:resource="ME2#ME2"/>
  <hdp:usesTo rdf:resource="BE2#BE2"/>
  <hdp:aspect>Structure</hdp:aspect>
  <pattern:name>User-centered Structure</pattern:name>
  <pattern:discussion>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</pattern:discussion>
  <hdp:usesTo rdf:resource="BE1#BE1"/>
  <pattern:context>reflejar la estructura de su dominio</pattern:context>
</hdp:HypermediaPattern>
```

[ME1]Hierarchical Organization

Contexto Se está definiendo una [AE1]User-centered Structure en la aplicación y se desea organizar grandes cantidades de información.

Problema ¿Cómo se debería organizar la información para ayudar al usuario a moverse con familiaridad?

Discusión Cuando se tiene que estructurar la información, se está intentando hacer explícito la representación del conocimiento que es inherente a la información. Debido a que se está usando dicha información para transmitir ciertas ideas, normalmente se suelen hacer explícitos ciertos subconjuntos del conocimiento. Las técnicas de estructuración básicas adoptadas universalmente dividen la información en bloques atómicos que normalmente contienen un elemento de información que pierde toda su utilidad si se divide en más partes [Lowe y Hall, 1999]. A la hora de organizar esos elementos de información, hay que tener en cuenta que los usuarios suelen pensar de manera diferente, e incluso un diseñador puede no pensar como los usuarios, por lo tanto es necesario pensar en la audiencia, en el contexto donde será utilizada la información, el lenguaje que usa la audiencia para describirla y la cantidad de información que se

presenta en un determinado momento. Por ejemplo para una frutería se puede tener organizadas las frutas por tropicales o locales, pero si los clientes están preocupados por la ecología, las frutas podrían estar organizadas por variedades tradicionales o transgénicas.

Ciertos tipos de conocimiento pueden ser estructurados como una jerarquía, por ejemplo, los libros a menudo son organizados en capítulos, que a su vez tienen secciones y subsecciones. Además, organizar la información por jerarquía de categorías puede ayudar al usuario a encontrar las cosas de manera más sencilla, ya que son muy familiares en la vida corporativa e institucional. Además, una organización jerárquica impone una disciplina útil a una aproximación analítica a los contenidos de la aplicación, ya que las jerarquías son sólo prácticas con material bien organizado.

Solución Dividir el espacio de información de la aplicación web en unidades lógicas. La mayoría de las "unidades" de información deberían ser ordenadas por importancia, y organizadas por el grado de interrelación entre ellas. Una vez que se ha determinado un conjunto lógico de prioridades, se puede construir una jerarquía desde los conceptos más importantes o más generales, bajando a los más específicos u opcionales. Usar nombres descriptivos para distinguir cada una de las categorías. Establecer las relaciones estructurales entre las unidades. La estructura jerárquica puede ser utilizada para retener la estructura original de la información contenida en la aplicación hipermedia.

Para relacionar los diferentes elementos de las jerarquía se pueden utilizar dos tipos de relaciones estructurales:

- Agregación. Es una relación de composición utilizada como mecanismo para referirse a un conjunto de nodos como a un todo. Por ejemplo, un libro es una agregación de capítulos.
- Generalización. Es una relación de inclusión que implica mecanismos de herencia. Por ejemplo, un libro es una generalización de conceptos como novela, poema o libro de texto.

Relacionado con Desde que una jerarquía puede tener varios niveles, se debe presentar al usuario cada subconjunto de unidades de información mediante el patrón `[BE1]Collection Center`. Para decidir la cantidad de información que debe contener cada nodo que conforma la estructura se puede consultar el patrón `[BE2]Node as a Single Unit`.

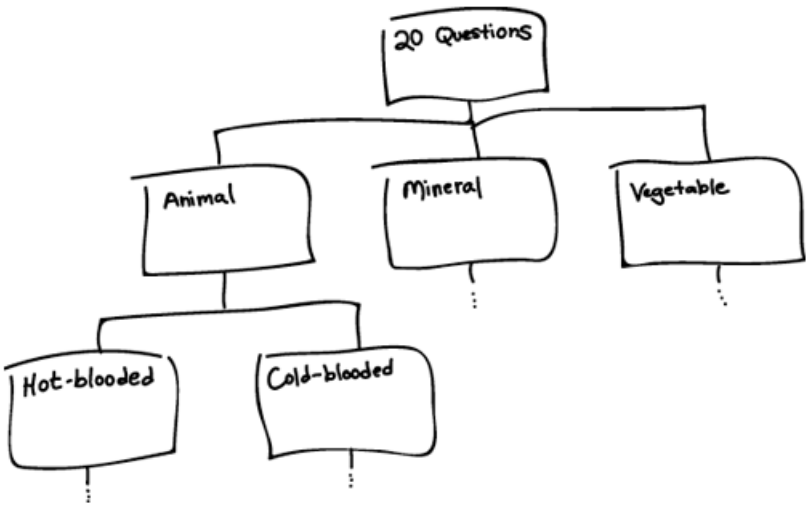


Figura A.3: Utilizar palabras que sean familiares a los usuarios y evitar que se desorienten con jerarquías profundas, no más de cuatro niveles(imagen tomada de [van Duyne *et al.*, 2002])

La estructura de la información afecta a la forma de navegación. Se puede utilizar el patrón `[MN1]Index Navigation` o el patrón `[MN2]Guided Tour Navigation` para hacer las jerarquías navegables.

Este patrón puede ser combinado con el patrón `[ME2]Task-based Organization`.

Si se desea mostrar cómo está organizada la información en su totalidad a la vez que se hace accesible, véase el patrón `[BN2]Site Map`. Si hay varios niveles en la jerarquía, véase el patrón `[BN1]Location Bread Crumbs` para mostrar al usuario el contexto de la página actual y ayudarle a entender la organización de la información. Además, puede ser necesario proporcionar herramientas de búsqueda, véase el patrón `[BN3]Search Action Module`.

Finalmente, se puede querer personalizar la estructura de la información según los diferentes usuarios de la aplicación, véase el patrón `[MZ2]Structure Personalization`.

Fuentes [van Duyne *et al.*, 2002],[31]

Formalización

```
<hdp:HypermediaPattern rdf:ID="ME1">
   <hdp:suggestsTo rdf:resource="ME2#ME2"/>
   <pattern:name>Hierarchical Organization</pattern:name>
   <hdp:usesTo rdf:resource="BE1#BE1"/>
   <pattern:context rdf:resource="AE1#AE1"/>
   <hdp:suggestsTo rdf:resource="BN2#BN2"/>
   <hdp:usesTo rdf:resource="BE2#BE2"/>
   <hdp:suggestsTo rdf:resource="BN3#BN3"/>
   <pattern:discussion>necesario pensar en la audiencia</pattern:discussion>
   <pattern:problem>moverse con familiaridad</pattern:problem>
   <hdp:suggestsTo rdf:resource="MN1#MN1"/>
   <hdp:level>Medium</hdp:level>
   <pattern:problem>
     <domain:CompositeComponent rdf:ID="información"/>
   </pattern:problem>
   <pattern:discussion>suelen pensar de manera diferente</pattern:discussion>
   <hdp:suggestsTo rdf:resource="MN2#MN2"/>
   <pattern:solution rdf:resource="#información"/>
   <pattern:solution>
     <domain:CompositeComponent rdf:about="#unidades lógicas"/>
   </pattern:solution>
   <hdp:suggestsTo rdf:resource="BN1#BN1"/>
   <hdp:aspect>Structure</hdp:aspect>
   <pattern:context>organizar grandes cantidades de información</pattern:context>
 </hdp:HypermediaPattern>
 <CompositeComponent rdf:ID="información">
   <hasComponents>
     <CompositeComponent rdf:ID="unidades logicas"/>
   </hasComponents>
   <relationship rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Generalization</relationship>
 </CompositeComponent>
</rdf:RDF>
```

[ME2]Task-based Organization

Contexto Se está definiendo una [AE1]User-centered Structure y se desea organizar las tareas del usuario.

Problema ¿Cómo organizar un conjunto de tareas relacionadas de tal manera que sean sencillas y rápidas de completar?

Discusión Cuando los usuarios tienen que realizar alguna tarea en la aplicación, como la compra de un producto, aprender, o cualquier otro proceso, la tarea es a menudo la primera de muchas otras. Si las tareas relacionadas no están organizadas de manera conjunta, los usuarios se pueden ver forzados a volver al punto de partida de la tarea para comenzar la siguiente, haciendo del uso de la aplicación algo tedioso. Completar múltiples tareas no es rápido ni sencillo si las tareas no están relacionadas.

Solución Estudiar como los usuarios realizan sus tareas, y la secuencia en que las llevan a cabo. Organizar las tareas según le gustaría al usuario utilizarlas. Entonces, agrupar

las tareas relacionadas utilizando una relación de composición para mostrar que esas tareas deben realizarse de manera conjunta, o están relacionadas.

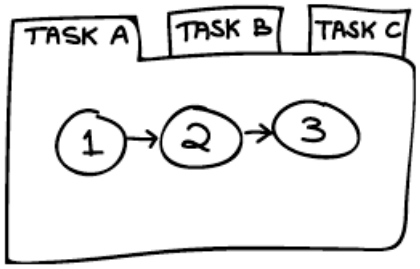


Figura A.4: Agrupar las tareas relacionadas de manera lógica transmitiendo al usuario que para alcanzar una determinada meta debe completar una serie de subtareas(imagen tomada de [van Duyne *et al.*, 2002])

Patrones relacionados Para decidir la cantidad de información que debe contener cada tarea se puede consultar el patrón `[BE2]Node as a Single Unit`.

Si existiesen muchas tareas que se pudiesen agrupar de manera jerárquica, véase el patrón `[ME1]Hierarchical Organization`. Para mostrar al usuario el orden en el que se deben realizar las tareas, véase el patrón `[MN2]Guided- Tour Navigation`.

El usuario debería estar informado sobre el estado de la interacción con la tarea que está llevando a cabo mediante el patrón `[MI2]Process Feed-back`. Para diferenciar los enlaces de navegación de los que representan acciones, véase el patrón `[BI1]Action Buttons`.

Fuentes [van Duyne *et al.*, 2002]

Formalización

```
<hdp:HypermediaPattern rdf:ID="ME2">
   <pattern:name>Task-based Organization</pattern:name>
   <pattern:discussion>uso de la aplicación algo tedioso</pattern:discussion>
   <hdp:level>Medium</hdp:level>
   <hdp:aspect>Structure</hdp:aspect>
   <pattern:context>organizar las tareas del usuario</pattern:context>
   <pattern:problem>sencillas y rápidas de completar</pattern:problem>
   <pattern:problem>
     <domain:CompositeComponent rdf:about="#conjunto de tareas"/>
   </pattern:problem>
   <pattern:discussion>Completar múltiples tareas no es rápido ni sencillo si las
    tareas no están relacionadas</pattern:discussion>
   <pattern:discussion>no están organizadas de manera conjunta, los usuarios se
```

```
    pueden ver forzados a volver al punto de partida</pattern:discussion>

   <hdp:suggestsTo rdf:resource="MI2#MI2"/>
   <pattern:context rdf:resource="AE1#AE1"/>
   <pattern:solution rdf:resource="#conjunto de tareas "/>
   <hdp:usesTo rdf:resource="BE2#BE2"/>
   <hdp:suggestsTo rdf:resource="ME1#ME1"/>
   <hdp:suggestsTo rdf:resource="MN2#MN2"/>
   <hdp:suggestsTo rdf:resource=" BI1# BI1"/>
 </hdp:HypermediaPattern>
 <CompositeComponent rdf:ID="conjunto de tareas">
   <hasComponents>
     <CompositeComponent rdf:ID="tareas"/>
   </hasComponents>
   <relationship rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   Aggregation</relationship>
 </CompositeComponent>
```

[BE1]Collection Center

Contexto Si la información ha sido organizada mediante [ME1]Hierarchical Organization y [ME2]Task-based Organization, con alguno de estos modos de navegación [MN1]Index Navigation o [MN2]Index Navigation, será necesario presentar al usuario cada tipo de información o tarea.

Problema ¿Cómo hacer llegar al usuario qué tipo de información o tarea recoge un conjunto de nodos para que sea entendible a la vez que puede seleccionar uno de ellos?

Discusión Cuando se está organizando el espacio de información, se tiende a dividirlo y a agruparlo en colecciones de cosas, como conceptos, cuadros, ciudades, personas, etc. Un asunto importante es cómo presentar al usuario los elementos que conforman una colección como un todo.

Además, una cuestión clave es cómo seleccionar, de cada miembro de la colección, los elementos de información que mejor lo describen con el objetivo de que llegue a ser un componente representativo de la colección, a la vez que proporcionan acceso a uno o más de sus componentes. La elección puede depender de varios factores: la naturaleza del objeto en sí mismo, el contexto proporcionado por la colección, el perfil del usuario, etc.

Solución Añadir un nodo que represente al conjunto de nodos de la colección y proporcione información adicional para mejorar la usabilidad y la efectividad de acceso a la información. Para una colección sencilla puede ser suficiente con proporcionar el título

de la colección y anclas expresivas para los miembros de la colección. Las anclas de los enlaces tienen un doble rol, describir el contenido de la colección y proporcionar un mecanismo de navegación. Si la colección es compleja, puede ser útil diseñar una página adicional dedicada a explicar el propósito, el fundamento y los antecedentes de la colección.

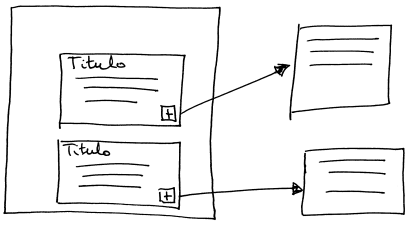


Figura A.5: Un nodo que sirva tanto de presentación como de acceso a un conjunto de unidades de información relacionadas

En resumen, un centro de colección podría incluir los siguientes elementos:

- una visión general del tema y propósito de la colección;
- una explicación explícita sobre sus escenarios de uso;
- información común a todos los miembros de la colección (p.e., si la colección es sobre el trabajo de un determinado artista, el centro puede incluir una introducción general a su actividad);
- anclas para los enlaces que permitan acceso a los miembros de la colección. Las anclas deberían identificar su destino por medio de iconos o etiquetas apropiadas o la combinación de los dos.
- el orden de las anclas dentro del centro debería también comunicar la topología de la colección, es decir el orden de los miembros dentro de ella.

Relacionado con El patrón `[BE3]Home Page` sería una especialización a nivel global de toda la aplicación.

Fuentes [Garzotto *et al.*, 1999], [12]

Formalización

```
<hdp:HypermediaPattern rdf:ID="BE1">
   <pattern:name>Collection Center</pattern:name>
   <hdp:level>Low</hdp:level>
```

```
    <hdp:aspect>Structure</hdp:aspect>
    <pattern:context>presentar al usuario cada tipo de información o
     tarea</pattern:context>
    <pattern:problem> hacer llegar al usuario qué tipo de información o tarea
     recoge un conjunto de nodos</pattern:problem>
    <pattern:problem>
      <domain:CompositeComponent rdf:about="#conjunto de nodos"/>
    </pattern:problem>
    <pattern:discussion>cómo seleccionar, de cada miembro de la
     colección</pattern:discussion>
    <pattern:discussion>cómo presentar al usuario los elementos que conforman una
     colección</pattern:discussion>
    <pattern:solution rdf:resource="#anclas"/>
    <pattern:solution rdf:resource="#nodo"/>
    <hdp:specializedBy rdf:resource="BE3#BE3"/>
  </hdp:HypermediaPattern>
  <domain:Anchor rdf:ID="anclas">
    <domain:location>
      <domain:Node rdf:ID="nodo">
        <domain:hasContents>
          <domain:Content rdf:about="#título de la colección"/>
        </domain:hasContents>
      </domain:Node>
    </domain:location>
  </domain:Anchor>
</rdf:RDF>
```

[BE2] Node as a Single Unit

Contexto La información está siendo organizada bien mediante [ME1]Hierarchical Organization o mediante [ME2]Task-based Organization, pero hay que decidir la extensión que debe tener cada nodo.

Problema ¿Cómo hacer que el usuario perciba que cada nodo es una unidad de información autocontenida y tiene sentido?

Motivación Las técnicas básicas de estructuración tienen en común romper la información en bloques atómicos, comúnmente llamados nodos. Muchas veces se encuentran nodos con mucha información que incluyen diferentes conceptos, los cuales no suelen ser relevantes para la tarea en curso, o a la inversa, un único concepto fragmentado en varios nodos que hace la lectura y la impresión más difícil para el lector que debe navegar de un fragmento al siguiente para poder ver completo el concepto.

Por lo tanto, un nodo debería incluir una unidad de información autocontenida que tenga sentido para un conjunto de usuarios que realizan una serie de tareas en un dominio dado y que pierde toda su utilidad si se divide en más.

A menudo, cuando la cantidad de información disponible sobre un tema es grande, es necesario balancear entre la cantidad de información presentada al usuario, la efectividad para obtener la información requerida con la cantidad de navegación involucrada en el proceso, y la sobrecarga cognitiva asociada.

Solución Incluir en un nodo toda la información relevante sobre un determinado tema, haciendo una clara separación entre la navegación a través de los diferentes temas y el modo en que son mostrados. Separar la evolución y la complejidad de la interfaz de la información.

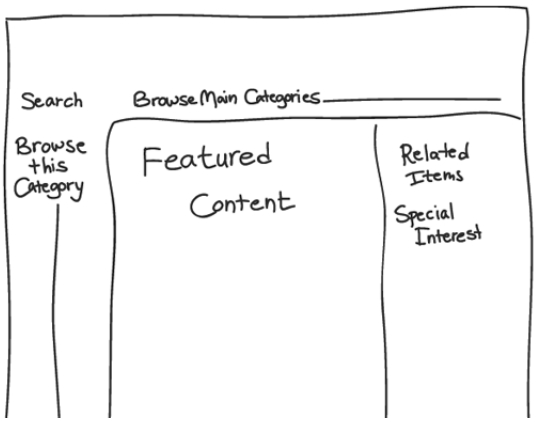


Figura A.6: Un nodo es la unidad fundamental de organización compuesta de contenidos relacionados y herramientas de navegación(imagen tomada de [Garrido *et al.*, 1997])

Relacionado con Si al final para que el nodo sea autocontenido existen demasiados elementos de información que no van a permitir una visualización clara, se puede utilizar el patrón `[MI1]Information on Demand` para que sea el usuario quien controle cuánta información quiere ver.

Para determinar cómo deberían estar organizados los contenidos de un nodo, véase el patrón `[AP1]Aesthetics`. Si el contenido quiere ser personalizado, véase el patrón `[MZ3]Content Personalization`.

Fuentes [Garrido *et al.*, 1997]

Formalización

```
<domain:Node rdf:ID="nodo">
   <domain:hasContents>
    <domain:Content rdf:ID="informacionRelevante"/>
```

```
    </domain:hasContents>
  </domain:Node>
<hdp:HypermediaPattern rdf:ID="BE2">
    <pattern:name>Node as a Single Unit</pattern:name>
    <hdp:level>Low</hdp:level>
    <hdp:aspect>Structure</hdp:aspect>
    <pattern:context>decidir la extensión que debe tener cada nodo</pattern:context>
    <pattern:problem>unidad de información autocontenida</pattern:problem>
    <pattern:problem rdf:resource="#nodo"/>
    <pattern:solution rdf:resource="#nodo"/>
    <hdp:suggestsTo rdf:resource="MZ3#MZ3"/>
    <hdp:suggestsTo rdf:resource="MI1#MI1"/>
    <hdp:suggestsTo rdf:resource="AP1#AP1"/>
  </hdp:HypermediaPattern>
```

[BE3]Home Page

Contexto Se ha definido una `[AE1]User-centered Structure` y es necesario contar con un punto de partida y de referencia para navegar a través del espacio de información.

Problema ¿Cómo hacer que el acceso a la aplicación sea atractivo al usuario mientras que se tiene en cuenta simultáneamente diferentes asuntos, incluyendo la navegación y contenidos?

Discusión Cuando un usuario accede por primera vez a una aplicación o sistema debería ser capaz de responder a las preguntas "¿Dónde estoy?" y "¿Qué hace esta aplicación?". La página principal o de entrada es normalmente el primer contacto que el usuario ve en la aplicación y en su diseño debe quedar claro el objetivo que tiene la aplicación. Si la página de inicio debe hacer hincapié en la imagen o una panorámica del sitio, las páginas interiores deben centrarse en los contenidos específicos que se quieran ofrecer.

También es el punto de entrada al esquema de navegación de la aplicación, el cual debería de ser fácil de utilizar y de entender. Los usuarios saben dos cosas: que algunos elementos pueden ser enlaces y que si ellos los seleccionan ocurrirá una acción, así que no deberían tener que diferenciar lo que es un enlace de lo que no es, a la vez que las acciones son claras y predecibles.

La página de inicio es la parte más representativa de una web y por lo tanto debe ser diseñada de una forma distinta al resto de las páginas. Obviamente las páginas de

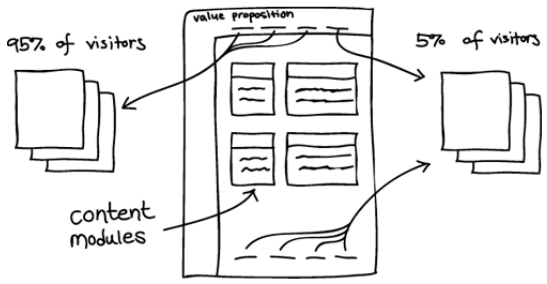


Figura A.7: Esboza una fuerte impresión de la aplicación con una página de inicio con títulos y logotipos convincentes y una navegación sencilla (imagen tomada de [van Duyne *et al.*, 2002])

inicio y las interiores debe compartir el mismo estilo, pero existen diferencias (v.g., la página de inicio no debería de tener un botón de inicio así misma).

Pero además, en las aplicaciones basadas en web, los usuarios no siempre entran por la página de inicio. Un sitio web es como una casa en la cual cada ventana es también una puerta, los usuarios pueden seguir enlaces desde motores de búsqueda y otros sitios web que llegan a nodos interiores de tu aplicación. Sin embargo, una de las primera cosas que hacen estos usuarios al llegar a tu sitio web es ir a la página de inicio. Claramente, a un sitio se puede acceder desde cualquier página, lo importante es hacer que la página de inicio sea un punto conocido al cual se pueda acceder con un simple clic.

Finalmente, también hay que tener en cuenta a aquellos usuarios que ya han utilizado la aplicación y no quieren desplazarse vínculo a vínculo hasta su destino. Es necesario destacar que es un error obligar a los usuarios a entrar en el sitio por la página de inicio. La llamada vinculación profunda permite que los usuarios entren al sitio en el punto exacto que sea de su interés.

Solución Crear una página inicial que al menos ofrezca estas tres características: el esquema de navegación de la aplicación a través de una lista de los niveles superiores de la jerarquía de información, un resumen de noticias o contenidos a destacar y opción de búsqueda. El espacio de está página inicial es limitado, por lo cual debe ser dividida entre los siguientes objetivos: crear el *look and feel* correcto, construyendo la identidad de la aplicación, proporcionar un contenido valioso, hacer la navegación fácil de usar, a la vez que se establece un diseño cohesivo para el resto de la aplicación.

Patrones relacionados Para determinar dónde colocar el esquema de navegación véase el patrón `[BP1]Navigation bar`. Como en las aplicaciones basadas en web se puede acceder a cualquier página, lo importante es hacer que la página de inicio sea un punto conocido a través de toda la organización de información mediante el patrón `[BN4]One Jump Home`.

Fuentes [van Duyne *et al.*, 2002], [33], [Nielsen, 2000]

Formalización

```
 <domain:CompositeComponent rdf:ID="Informacion"/>
 <domain:Hyperdocument rdf:ID="aplicacion"/>
 <domain:Content rdf:ID="informacionRelevante"/>
 <domain:Node rdf:ID="EsquemaNavegacion">
   <domain:hasContents rdf:resource="#informacionRelevante"/>
 </domain:Node>

<hdp:HypermediaPattern rdf:ID="BE3">
    <pattern:discussion>es el punto de entrada al esquema de
    navegación</pattern:discussion>
    <hdp:aspect>Structure</hdp:aspect>
    <hdp:level>Low</hdp:level>
    <pattern:problem>incluyendo la navegación y contenidos</pattern:problem>
    <pattern:name>Home Page</pattern:name>
    <pattern:discussion> ¿Dónde estoy? y ¿Qué hace esta
    aplicación?</pattern:discussion>
    <pattern:context>contar con un punto de partida y de referencia para navegar a
    través del espacio de información</pattern:context>
    <pattern:discussion>el primer contacto que el usuario ve</pattern:discussion>
    <pattern:problem>el acceso a la aplicación sea atractivo al
    usuario</pattern:problem>
    <pattern:problem rdf:resource="#aplicacion"/>
    <pattern:solution rdf:resource="#Informacion"/>
    <pattern:solution rdf:resource="#EsquemaNavegacion"/>
    <pattern:solution>
      <domain:Root rdf:ID="paginaInicial">
        <domain:hasComponents rdf:resource="#Informacion"/>
        <domain:relationship rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        aggregation</domain:relationship>
        <domain:hasComponents rdf:resource="#EsquemaNavegacion"/>
      </domain:Root>
    </pattern:solution>
    <hdp:usesTo rdf:resource="BP1#BP1"/>
    <hdp:suggestsTo rdf:resource="BN4#BN4"/>
  </hdp:HypermediaPattern>
```

A.1.2. [AN1]Multiple Ways to Navigate

Ahora que sabes qué tipo de información y operaciones están esperando los usuarios de tu aplicación, necesitas guiarles tanto a la hora de explorar el espacio de información, como

a la de acometer sus tareas o encontrar lo que buscan. El conjunto de patrones aquí presentados bajo el nombre `Multiple Ways to Navigate` recoge soluciones a los principales problemas de navegación de las aplicaciones hipermedia: explorar el espacio de navegación y encontrar los caminos a la información deseada, sin sufrir desorientación o sobrecarga cognitiva.

`[AN1]Multiple Ways to Navigate`

Contexto Se está construyendo una `[A]Hypermedia Application` y la información ha sido organizada siguiendo una `[AE1]User-centered Structure`. Ahora, es necesario dotar al usuario de mecanismos de navegación y de acceso al espacio de información.

Problema ¿Cómo hacer que el usuario pueda moverse por la información de diferentes modos, según su motivación o intención de manera clara y consistente a la vez que se le facilita encontrar la información deseada?

Discusión 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? [Nielsen, 2000]:

- 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. Quizás 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.

Y a veces, los usuarios no quieren perder el tiempo aprendiendo a navegar por el sistema, sólo quieren entrar, conseguir la información y salir tan rápido como sea posible.

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 grandes 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.

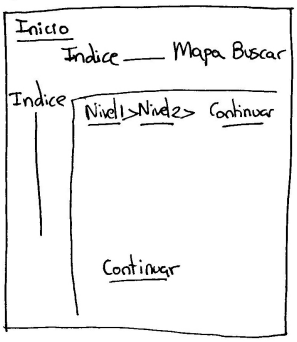


Figura A.8: Proporcionar a los usuarios diferentes modos de navegar por el sistema para que utilicen la que mejor se adapte a sus metas y deseos

Relacionado con El patrón `[MN1]Index Navigation` ayuda a los usuarios a seleccionar un destino, mientras que el patrón `[MN2]Guided Tour Navigation` 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]Navigational Context`. 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]Location Bread Crumbs. Para mostrar la situación del usuario con respecto al espacio de información total de la aplicación, utilizar el patrón [BN2]Site Map. Para que el usuario pueda conseguir la información deseada de forma directa, se puede proporcionar un [BN3]Search Action Module. Los usuarios deberían ser capaces de volver a la página principal o a otros puntos mayores de navegación en un sólo paso con [BN4]One Jump Home.

Fuentes [van Duyne *et al.*, 2002]

Formalización

```
<domain:CompositeComponent rdf:ID="Informacion"/>
<hdp:HypermediaPattern rdf:ID="AN1">
   <hdp:aspect>Navigation</hdp:aspect>
   <pattern:discussion> ofrecer al usuario un soporte de navegación </pattern:discussion>
   <pattern:context>dotar al usuario de mecanismos de navegación y de acceso al
    espacio de información</pattern:context>
   <pattern:problem>pueda moverse por la información de diferentes
    modos</pattern:problem>
   <hdp:usesTo rdf:resource="BN3#BN3"/>
   <hdp:usesTo rdf:resource="BN1#BN1"/>
   <pattern:context rdf:resource="A#A"/>
   <pattern:discussion>espacio es muy grande, y la navegación
    difícil</pattern:discussion>
   <pattern:discussion>respuestas claras y no ambiguas a dos cuestiones básicas:
    ¿A dónde puedo ir?
    y ¿Dónde estoy?</pattern:discussion>
   <hdp:usesTo rdf:resource="BN2#BN2"/>
   <pattern:problem> facilita encontrar la información deseada</pattern:problem>
   <pattern:problem rdf:resource="#Informacion"/>
   <pattern:solution rdf:resource="#Informacion"/>
   <hdp:usesTo rdf:resource="BN4#BN4"/>
   <hdp:usesTo rdf:resource="MN3#MN3"/>
   <hdp:usesTo rdf:resource="MN2#MN2"/>
   <hdp:level>High</hdp:level>
   <pattern:name>Multiple Ways to Navigate</pattern:name>
   <pattern:discussion>usuarios suelen tener diferentes
    intenciones</pattern:discussion>
   <pattern:context rdf:resource="AE1#AE1"/>
   <hdp:usesTo rdf:resource="MN1#MN1"/>
 </hdp:HypermediaPattern>
```

[MN1]Index Navigation

Contexto La información ha sido organizada mediante [AE1]Hierarchical Organization y es necesario poder acceder a ella.

Problema ¿Cómo proporcionar acceso directo a un determinado miembro de un grupo de nodos?

Discusión Este requisito es menos obvio de lo que pueda parecer a primera vista; de hecho, en muchas situaciones, el usuario no es capaz de hacer una elección ya que pueden no conocer suficientemente bien el tema de la aplicación, no ser capaz de identificar los elementos en la lista, o no tener una meta específica en mente. Además uno de los inconvenientes de la navegación puede ser la necesidad de volver a un punto de partida para acceder a otro nodo, necesitando así dos pasos para moverse de un nodo a otro.

Por lo tanto, es necesario hacer accesible la información al usuario para que pueda decidir.

Solución Definir enlaces desde el punto de entrada al conjunto de nodos a cada uno de sus miembros, y de cada miembro al punto de entrada. Para acelerar la navegación, mantener siempre accesible el punto de entrada que contiene los enlaces a cada miembro para que el usuario pueda acceder directamente a cualquiera de ellos sin tener que volver al punto de entrada.

El punto de entrada o índice es una lista de anclas, enlaces o nodos ordenados por el alfabeto, el tema, el autor, la materia, etc. Esta definición incluye herramientas comunes en papel como índices alfabéticos, tabla de contenidos, glosarios y otros.

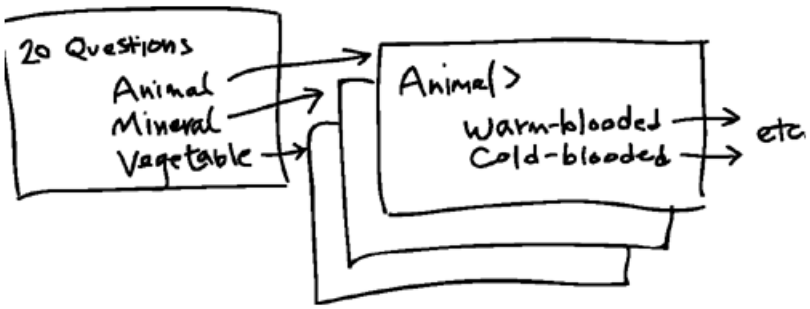


Figura A.9: Presentar los elementos en un formato simple y explorable dando a los usuarios una clara idea hacia dónde pueden ir a la vez que aprenden la estructura organizativa de la aplicación(imagen tomada de [van Duyne *et al.*, 2002])

Relacionado con El patrón `[BE1]Collection Center` define qué contenidos debería contener el punto de entrada o índice. Para ayudar a los usuarios a moverse por los elementos de la colección siguiendo algún orden se puede utilizar el patrón `[MN2]Guided Tour Navigation`.

Además si este patrón ha sido utilizado de manera recursiva para acceder al espacio de información de manera jerárquica, es necesario utilizar el patrón `[BN1]Location Bread Crumbs` para orientar al usuario sobre dónde se encuentra.

Para adaptar este mecanismo de navegación a diferentes contextos, utilizar `[MN3]Navigational Context`.

Para ayudar al usuario a diferenciar entre los elementos de navegación y la información véase el patrón `[MP1]Information-Interaction Decoupling`.

Fuentes [Garzotto *et al.*, 1999], [12]

Formalización

```
<domain:Anchor rdf:ID="entrada">
   <domain:location>
     <domain:Node rdf:ID="puntoentrada">
       <hasAnchors>
         <domain:Anchor rdf:ID="anclas"/>
       </hasAnchors>
     </domain:Node>
   </domain:location>
 </domain:Anchor>
<domain:CompositeComponent rdf:ID="gruponodos"/>
 <domain:Anchor rdf:ID="miembro">
   <domain:location rdf:resource="#gruponodos"/>
 </domain:Anchor>
 <domain:Link rdf:ID="enlaces">
   <domain:hasTarget rdf:resource="#miembro"/>
   <domain:hasSource rdf:resource="#entrada"/>
 </domain:Link>
 <hdp:HypermediaPattern rdf:ID="MN1">
   <hdp:level>Medium</hdp:level>
   <hdp:aspect>Navigation</hdp:aspect>
   <pattern:context>poder acceder</pattern:context>
   <pattern:context rdf:resource="AE1#AE1"/>
   <pattern:context>información</pattern:context>
   <pattern:problem>proporcionar acceso directo</pattern:problem>
   <pattern:problem rdf:resource="#gruponodos"/>
   <pattern:solution rdf:resource="#enlaces"/>
   <pattern:solution rdf:resource="#gruponodos"/>
   <pattern:solution rdf:resource="#puntoentrada"/>
   <hdp:suggestsTo rdf:resource="MP1#MP1"/>
   <hdp:suggestsTo rdf:resource="BN1#BN1"/>
   <hdp:usesTo rdf:resource="BE1#BE1"/>
   <hdp:specializedBy rdf:resource="MN3#MN3"/>
   <hdp:suggestsTo rdf:resource="MN2#MN2"/>
 </hdp:HypermediaPattern>
```

`[MN2]Guided Tour Navigation`

Contexto La información ha sido organizada siguiendo `[BE1]Hierarchical Organization` o `[BE2]Task-based Organization` y se desea guiar al usuario en el modo de explorar un conjunto de nodos.

Problema ¿Cómo proporcionar un acceso fácil a un grupo pequeño de nodos relacionados, asumiendo que el usuario no tiene razones para seleccionar uno de ellos?

Discusión Algunos usuarios con poco conocimiento del dominio de la aplicación o que acceden por primera vez, necesitan a veces adquirir conocimiento del contenido de la aplicación o quieren conseguir una visita rápida por la aplicación, más que lanzarse a navegar sin sentido.

Solución Identificar un orden entre el conjunto de nodos y crear enlaces secuenciales entre ellos. Los enlaces pueden ser uni o bi-direccionales. Una variante es un visita guiada circular, donde el último nodo es enlazado al primero o a un nodo superior de la estructura para salir del ciclo. Es necesaria un ancla de acceso al primer nodo de la colección.

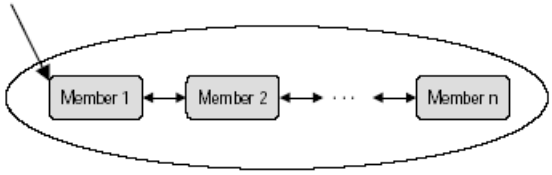


Figura A.10: Proporcionar un itinerario a aquellos usuarios que acceden por primera vez a la aplicación (imagen tomada de [Garzotto *et al.*, 1999])

Relacionado con Utilizar el patrón `[BE1]Collection Center` para definir el punto de entrada. Este patrón puede combinarse con `[MN1]Index Navigation` para salir del itinerario al final del recorrido o en cualquier momento.

Para adaptar este mecanismo de navegación a diferentes contextos, utilizar `[MN3]Navigational Context`.

Fuentes [Garzotto *et al.*, 1999], [12]

Formalización

```
<domain:Anchor rdf:ID="entrada">
    <domain:location>
      <domain:CompositeComponent rdf:ID="grupopequeñosnodos"/>
    </domain:location>
  </domain:Anchor>
  <domain:Anchor rdf:ID="siguiente">
    <domain:location rdf:resource="#grupopequeñosnodos"/>
  </domain:Anchor>
<domain:Link rdf:ID="enlaces">
```

```
    <domain:hasSource rdf:resource="#entrada"/>
    <domain:hasTarget rdf:resource="#siguiente"/>
  </domain:Link>
 <hdp:HypermediaPattern rdf:ID="MN2">
   <pattern:name>Guided Tour Navigation</pattern:name>
    <hdp:level>Medium</hdp:level>
    <hdp:aspect>Navigation</hdp:aspect>
    <pattern:context rdf:resource="BE1#BE1"/>
    <pattern:context rdf:resource="BE2#BE2"/>
    <pattern:context>guiar al usuario en el modo de explorar un conjunto de
     nodos</pattern:context>
    <pattern:problem> el usuario no tiene razones para seleccionar uno de
     ellos</pattern:problem>
    <pattern:problem> acceso fácil a un grupo pequeño de nodos</pattern:problem>
    <pattern:problem rdf:resource="#grupopequeñosnodos"/>
    <pattern:discussion> necesitan a veces adquirir conocimiento del contenido de
     la aplicación</pattern:discussion>
    <pattern:discussion>onseguir una visita rápida por la
     aplicación</pattern:discussion>
    <pattern:solution rdf:resource="#grupopequeñosnodos"/>
    <pattern:solution rdf:resource="#enlaces"/>
    <hdp:suggestsTo rdf:resource="MN1#MN1"/>
    <hdp:usesTo rdf:resource="BE1#BE1"/>
    <hdp:specializedBy rdf:resource="MN3#MN3"/>
  </hdp:HypermediaPattern>
```

[MN3]Navigational Context

Contexto Se ha utilizado `[MN1]Index Navigation` o `[MN2]Guided Tour Navigation` como estilo de navegación, pero se desea que la navegación dependa del contexto de la información.

Problema ¿Cómo proporcionar al usuario diferentes modos de explorar un nodo según el estado actual de la navegación?

Discusión Normalmente las aplicaciones tratan con colecciones de cosas, como conceptos, cuadros, ciudades, personas, etc. Estas colecciones pueden ser exploradas de diferentes modos, según la tarea que el usuario esté realizando. Por ejemplo, se pueden querer explorar los libros de un autor, los libros de un cierto periodo de tiempo o movimiento literario, etc., y es deseable dar al usuario diferentes tipos de respuesta en diferentes contextos, mientras que se puede mover fácilmente de nodo a nodo.

Un ejemplo sería que en una enciclopedia sobre inventores e inventos, lleguemos a “Thomas A. Edison” y a la invención de la “bombilla”. Sin embargo, también podemos llegar a la bombilla mientras exploramos los inventos durante un específico periodo de tiempo, siguiendo una historia de los más famosos inventos, o mientras visitamos otro invento como la “luz fluorescente” y seguimos el enlace a “invenciones

relacionadas". Es obvio que estamos explorando el mismo nodo bajo tres diferentes perspectivas: como un invento de Thomas A. Edison, como un invento del siglo XIX, y como un invento relacionado con otros como la lámpara fluorescente.

Esto significa que se necesita no sólo presentar la información en diferentes modos en cada uno de los casos anteriores, sino también proporcionar diferentes enlaces o índices. Por lo tanto, esos nodos deberían ser conectados con enlaces contextuales, por ejemplo sería deseable poder movernos al siguiente invento de Thomas A. Edison, o al siguiente invento del siglo XIX. Sin embargo, aunque el contexto en el cual un nodo está siendo accedido no concierne al nodo, es impracticable y puede provocar inconsistencias tener diferentes objetos que representen al mismo componentes para cada diferente contexto.

Solución Separar la información del contexto en el cual se explora. Definir enlaces contextuales (conectan objetos en un contexto) entre nodos relacionados. Mostrar sólo la información relacionada con el contexto en el cual es visitado el nodo incluyendo información adicional sobre el contexto y anclas adicionales para los enlaces contextuales.

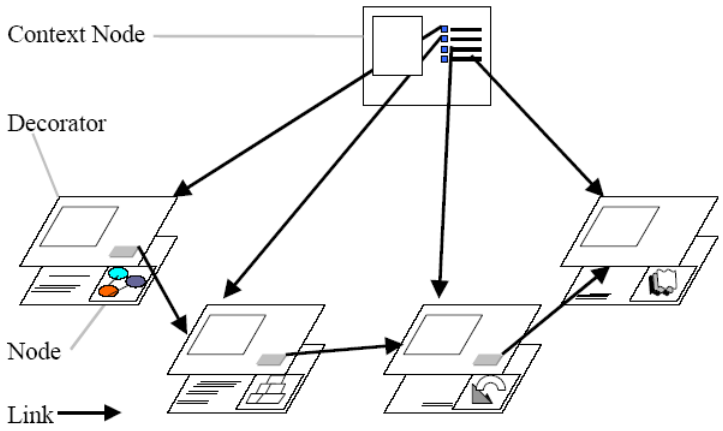


Figura A.11: Adaptar la navegación a los diferentes contextos en los que puede ser explorada la información (imagen tomada de [Garrido *et al.*, 1997])

Relacionado con :

La unidad de información, independientemente del contexto, debería ser definida siguiendo el patrón `[BE2]Node as a Single Unit`.

Además, si este patrón ha sido utilizado de manera recursiva para acceder al espacio de información de manera jerárquica, utilizar el patrón `[BN1]Location Bread Crumbs` para orientar al usuario sobre dónde se encuentra.

Fuentes [Garrido *et al.*, 1997]

Formalización

```
<domain:Anchor rdf:ID="siguiente">
    <domain:location>
     <domain:Node rdf:ID="nodo">
     <domain:hasContents rdf:resource="#informacionContexto"/>
     </domain:Node>
    </domain:location>
  </domain:Anchor>
<domain:ContextualLink rdf:ID="enlaceContextual">
  <domain:context rdf:resource="#informacionContexto"/>
  <domain:hasTarget rdf:resource="#siguiente"/>
</domain:ContextualLink>
<hdp:HypermediaPattern rdf:ID="MN3">
    <pattern:name>Navigational Context</pattern:name>
    <hdp:aspect>Navigation</hdp:aspect>
    <hdp:level>Medium</hdp:level>
    <pattern:discussion>no sólo presentar la información en diferentes
     modos</pattern:discussion>
    <pattern:context>a navegación dependa del contexto de la
     información</pattern:context>
    <pattern:discussion>según la tarea que el usuario esté
     realizando</pattern:discussion>
    <pattern:discussion>roporcionar diferentes enlaces o
     índices</pattern:discussion>
    <pattern:discussion> ser exploradas de diferentes modos</pattern:discussion>
    <pattern:context rdf:resource="MN1#MN1"/>
    <pattern:context rdf:resource="MN2#MN2"/>
    <pattern:problem>diferentes modos de explorar un nodo según el estado actual
     de la navegación</pattern:problem>
     <pattern:problem rdf:resource="#nodo"/>
     <pattern:solution rdf:resource="#enlaceContextual"/>
    <hdp:usesTo rdf:resource="BE2#BE2"/>
    <hdp:usesTo rdf:resource="BN1#BN1"/>
    <pattern:discussion> colecciones de cosas</pattern:discussion>
  </hdp:HypermediaPattern>
```

`[BN1]Location Bread Crumbs`

Contexto La información ha sido organizada siguiendo alguno de los esquemas presentados en `[AE1]User-centered Structure` y se están definiendo `[AN1]Multiple Way to Navigate` para acceder a la información, pero el usuario necesita conocer el estado actual de la navegación.

Problema ¿Cómo se puede proporcionar un indicador de situación del estado actual de la navegación, combinando una herramienta de orientación con un modo sencillo

para navegar a un conjunto de nodos relacionados, en el mismo nivel o más alto de abstracción?

Discusión En muchas aplicaciones hipermedia, particularmente en aquellas que involucran estructuras espaciales o de tiempo, es necesario proporcionar al lector un modo de entender dónde está. Es fácil que el usuario se sienta perdido al perder el rastro de dónde se encuentra con relación a otras páginas de la aplicación. Además, si el usuario accede a una determinada página a través de un marcador, una URL, o desde un motor de búsqueda, necesitará ayuda para orientarse por si mismo.

Solución Proporcionar un objeto navegacional, conocido como *bread crumbs* que actúe como un índice a otros nodos o subíndices. Este objeto se mantiene siempre visible junto con los objetos destino.

El término *bread crumbs* (migas de pan) hace referencia a la barra de arriba que aparece en las páginas web mostrando el rastro de páginas que un usuario lleva desde la página principal a la página actual (ver figura A.12). Cada página en la barra es un enlace que permite rápidamente dar marcha atrás. Los *bread crumbs* además permiten al usuario ver en qué lugar se encuentra en relación a la página principal, proporcionando información sobre la estructura del sitio web que está visitando.

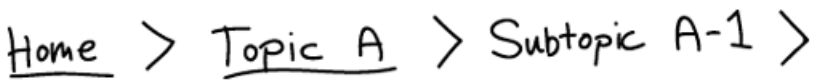


Figura A.12: Mostrar el rastro seguido desde la página principal hasta la página actual ayudando al usuario a saber donde está dentro de una determinada categoría (imagen tomada de [van Duyne *et al.*, 2002])

Relacionado con : Para proporcionar orientación al usuario a nivel global de la aplicación véase el patrón `[BP1]Navigation Bar`.

Fuentes [van Duyne *et al.*, 2002], [Garrido *et al.*, 1997]

Formalización

```
 <domain:Anchor rdf:ID="anclas">
    <domain:location>
     <domain:Content rdf:ID="indicadorsituacion"/>
   </domain:location>
  </domain:Anchor>
<domain:Link rdf:ID="indice">
```

```
    <domain:hasSource rdf:resource="#anclas"/>
  </domain:Link>
<domain:Node rdf:ID="nodo">
   <hasAnchors rdf:resource="#anclas"/>
   <domain:hasContents rdf:resource="#indicadorsituacion"/>
 </domain:Node>
 <hdp:HypermediaPattern rdf:ID="BN1">
    <pattern:name>Location Bread Crumbs</pattern:name>
    <hdp:level>Low</hdp:level>
    <hdp:aspect>Navigation</hdp:aspect>
    <pattern:context rdf:resource="AN1#AN1"/>
    <pattern:context rdf:resource="AE1#AE1"/>
    <pattern:discussion>el usuario se sienta perdido al perder el rastro de dónde
     se encuentra</pattern:discussion>
    <pattern:problem>proporcionar un indicador de situación  del estado actual de
     la navegación</pattern:problem>
    <pattern:discussion>ayuda para orientarse por si mismo</pattern:discussion>
    <pattern:discussion>proporcionar al lector un modo de entender dónde
     está</pattern:discussion>
    <pattern:context>conocer el estado actual de la navegación</pattern:context>
    <pattern:problem rdf:resource="#indicadorsituacion"/>
    <pattern:solution rdf:resource="#nodo"/>
    <pattern:solution rdf:resource="#indice"/>
    <hdp:usesTo rdf:resource="BP1#BP1"/>
  </hdp:HypermediaPattern>
```

[BN2]Site Map

Contexto Se está proporcionando [AN1]Multiple Ways to Navigate sobre [AE1]User-centered Structure, y se desea que el usuario sea capaz de orientarse dentro del espacio total de información.

Problema ¿Cómo mostrar al usuario dónde se encuentra globalmente con respecto a la estructura de la aplicación?

Discusión Normalmente cuando la aplicación tiene más de dos niveles en la estructura jerárquica y muchos elementos en cada nivel, el usuario puede llegar a sentirse perdido o confuso de su situación actual dentro del espacio de información global de la aplicación. Por lo tanto es necesario ayudarle a encontrar su camino.

Los usuarios quieren saber dónde pueden ir o cómo está organizada la aplicación o ver donde están.

Solución Crear un mapa de la aplicación mostrando la estructura jerárquica de la aplicación en un nodo a parte. El mapa muestra todos los elementos de cada nivel y al menos los correspondientes a la navegación global de la aplicación. La composición del mapa suele ser en forma de árbol. El mapa es accesible desde cada página de la aplicación y la página desde la cual se está accediendo al mapa está destaca en él.

Un mapa es como una tabla de contenidos de la aplicación que muestra todas las páginas disponibles al usuario. Además, responde las preguntas "¿Dónde estoy?" y "¿Qué hay disponible?". También permite a los usuarios acceder a una determinada página de forma directa.

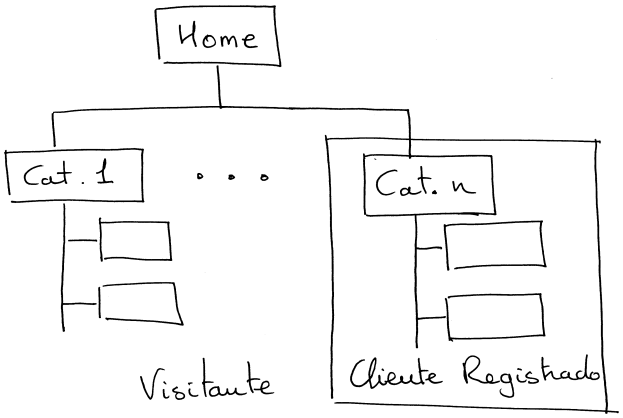


Figura A.13: Etiquetar y colorear cada nivel para indicar las diferentes categorías de la estructura e incluso las áreas que son de acceso restringido

Relacionado con : Para que el mapa sea accesible desde todas las páginas de la aplicación hay que posicionarlo en el patrón `[BP1]Navigation Bar`.

Fuentes [33]

Formalización

```
<domain:Anchor rdf:ID="anclas">
    <domain:location>
      <domain:Node rdf:ID="mapa">
        <hasAnchors rdf:resource="#anclas"/>
      </domain:Node>
    </domain:location>
  </domain:Anchor>
<domain:Link rdf:ID="indice">
    <domain:hasSource rdf:resource="#anclas"/>
    <domain:hasTarget>
      <domain:Anchor rdf:ID="pagina">
        <domain:location>
          <domain:Root rdf:ID="estructuraaplicacion">
            <domain:relationship
             rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
             >aggregation</domain:relationship>
            <domain:hasComponents rdf:resource="#mapa"/>
          </domain:Root>
        </domain:location>
      </domain:Anchor>
    </domain:hasTarget>
  </domain:Link>
```

```
<hdp:HypermediaPattern rdf:ID="BN2">
   <pattern:name>Site Map</pattern:name>
   <hdp:level>Low</hdp:level>
   <hdp:aspect>Navigation</hdp:aspect>
   <pattern:context rdf:resource="AN1#AN1"/>
   <pattern:context rdf:resource="AE1#AE1"/>
   <pattern:context> sea capaz de orientarse dentro del espacio total de
    información</pattern:context>
   <pattern:problem>dónde se encuentra globalmente con respecto a la estructura
    de la aplicación</pattern:problem>
   <pattern:discussion>quieren saber dónde pueden ir o cómo está organizada la
    aplicación o ver donde están</pattern:discussion>
   <pattern:discussion>sentirse perdido o confuso</pattern:discussion>
   <pattern:discussion>la aplicación tiene más de dos niveles en la estructura
    jerárquica y muchos elementos en cada nivel</pattern:discussion>
   <pattern:solution rdf:resource="#mapa"/>
   <pattern:problem rdf:resource="#estructuraaplicacion"/>
   <pattern:solution rdf:resource="#indice"/>
   <hdp:suggestsTo rdf:resource="BP1#BP1"/>
 </hdp:HypermediaPattern>
```

[BN3]Search Action Module

Contexto Se están definiendo [AN1]Multiple Ways to Navigate pero a veces los usuarios quieren encontrar rápidamente una información concreta.

Problema ¿Cómo ayudar a los usuarios a acceder de manera directa a una determina información?

Discusión Existen ciertos usuarios que saben exactamente lo que están buscando. Además, la web ofrece un gran espacio de información que crece a gran velocidad, sufre de cambios, y donde la información es añadida, borrada, clasificada y reorganizada, con criterios altamente heterogéneos. Por lo tanto, es necesario ayudar a los usuarios a encontrar un determinado asunto en tal cantidad de datos.

Solución Ofrecer un módulo de búsqueda accesible en cada nodo, usando una simple frase que indique el espacio de búsqueda para escribir las palabras claves y un botón para comenzar la búsqueda. Si la aplicación contiene mucha información, añadir una lista de subsecciones y la palabra para indicar la cadena de caracteres a buscar.

Para mejorar la capacidad de búsqueda hay que delegar el servicio a motores más sofisticados, que son capaces de producir una respuesta estructurada con más detalle o información mejor formateada.

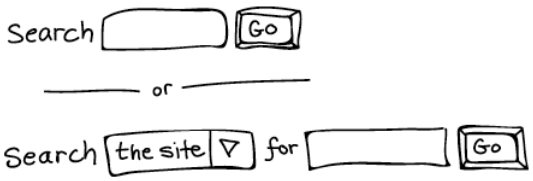


Figura A.14: Los usuarios utilizarán la herramienta de búsqueda si la mantienes simple y aparece en cada página (imagen tomada de [van Duyne *et al.*, 2002])

Relacionado con : Para que la herramienta de búsqueda sea accesible desde todas las páginas de la aplicación hay que posicionarla en el patrón `[BP1]Navigation Bar`. El botón que inicia la búsqueda debería ser reflejado utilizando el patrón `[BI1]Action Button`.

Fuentes [van Duyne *et al.*, 2002], [33]

Formalización

```
<domain:Content rdf:ID="boton"/>
<domain:Content rdf:ID="espaciobusqueda"/>
<domain:Root rdf:ID="estructuraaplicacion">
    <domain:hasComponents>
      <domain:Node rdf:ID="modulobusqueda">
           <domain:hasContents rdf:resource="#boton"/>
           <domain:hasContents>
           <domain:Content rdf:ID="espaciobusqueda"/>
           </domain:hasContents>
      </domain:Node>
     </domain:hasComponents>
     <domain:relationship rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >aggregation</domain:relationship>
</domain:Root>
<hdp:HypermediaPattern rdf:ID="BN3">
    <hdp:level>Low</hdp:level>
    <hdp:aspect>Navigation</hdp:aspect>
    <pattern:name>Search Action Module</pattern:name>
    <pattern:context rdf:resource="AN1#AN1"/>
    <pattern:context>encontrar rápidamente una información
     concreta</pattern:context>
    <pattern:discussion>ayudar a los usuarios a encontrar un determinado asunto en
     tal cantidad de datos</pattern:discussion>
    <pattern:discussion>usuarios que saben exactamente lo que están
     buscando</pattern:discussion>
    <pattern:problem> acceder de manera directa a una determina
     información</pattern:problem>
    <pattern:problem rdf:resource="#informacion"/>
    <pattern:solution rdf:resource="#boton"/>
    <pattern:solution rdf:resource="#estructuraaplicacion"/>
    <pattern:solution rdf:resource="#espaciobusqueda"/>
    <hdp:usesTo rdf:resource="BP1#BP1"/>
    <hdp:usesTo rdf:resource="BI1#BI1"/>
  </hdp:HypermediaPattern>
```

[BN4]One Jump Home

Contexto Se está definiendo [AN1] Multiple Ways to Navigate, pero los usuarios necesitan volver a un sitio seguro y familiar.

Problema ¿Cómo definir un modo que permita, en cualquier momento, volver al inicio de la aplicación?

Motivación A veces los usuarios navegan de manera arbitraria por la aplicación si un rumbo fijo, pero llega un momento en el que desean volver al punto de partida. Además los motores de búsqueda y los usuarios almacenan URLs que enlazan con nodos internos de la aplicación web, lo que dificulta saber dónde estás.

Solución Usar un contenido fijo, como el logo de la aplicación, una etiqueta de texto o el icono de una casa, como enlace a la página principal. Añadirlo en cada página y situarlo arriba de la página y si es apropiado también al final de ella. Asegurarse de que el enlace está siempre accesible en el mismo sitio.

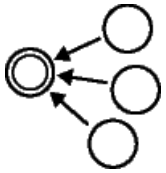


Figura A.15: La página principal suele ser el punto de partida de la interacción (imagen tomada de [21])

Relacionado con : Para definir los elementos que contendrá la página principal de la aplicación véase el patrón [BE3]Home Page.

Fuentes [21]

Formalización

```
<domain:Anchor rdf:ID="entrada">
    <domain:location rdf:resource="#paginaprincipal"/>
  </domain:Anchor>
  <domain:Content rdf:ID="volverinicio"/>
  <domain:Anchor rdf:ID="anclas">
    <domain:location rdf:resource="#volverinicio"/>
  </domain:Anchor>
  <domain:Link rdf:ID="indice">
    <domain:hasTarget rdf:resource="#entrada"/>
    <domain:hasSource rdf:resource="#anclas"/>
```

```
  </domain:Link>
<hdp:HypermediaPattern rdf:ID="BN4">
 <hdp:level>Low</hdp:level>
    <pattern:discussion>desean volver al punto de partida</pattern:discussion>
    <pattern:discussion>los usuarios navegan de manera
     arbitraria</pattern:discussion>
    <pattern:context> los usuarios necesitan volver a un sitio seguro y
     familiar</pattern:context>
    <pattern:name>One Jump Home</pattern:name>
    <hdp:suggestsTo rdf:resource="BE3#BE3"/>
    <pattern:discussion>sin un rumbo fijo</pattern:discussion>
    <pattern:context rdf:resource="AN1#AN1"/>
    <hdp:aspect>Navigation</hdp:aspect>
    <pattern:problem>volver al inicio de la aplicación</pattern:problem>
    <pattern:solution rdf:resource="#volverinicio"/>
    <pattern:problem rdf:resource="#paginaprincipal"/>
    <pattern:solution rdf:resource="#paginaprincipal"/>
    <pattern:solution rdf:resource="#indice"/>
    <pattern:solution>
      <domain:Node rdf:ID="pagina">
        <hasAnchors rdf:resource="#anclas"/>
        <domain:hasContents rdf:resource="#volverinicio"/>
      </domain:Node>
    </pattern:solution>
  </hdp:HypermediaPattern>
```

A.1.3. [AP1]Aesthetics

Los patrones aquí presentados bajo la categoría `Aesthetics` son abstractos e independientes del entorno utilizado para la implementación de la interfaz e involucran la combinación correcta de los elementos, tanto en su número como en sus relaciones espaciales y temporales, de tal manera que dichos elementos sean presentados de una manera efectiva.

[AP1]Aesthetics

Contexto Se está diseñando una `[A]Hypermedia Application` y la información ha sido organizada mediante unidades lógicas según una `[AE1]User-centered Structure` a la vez que se han proporcionado `[AN1]Multiple ways to navegate`. 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 nodos son mostrados de manera estética?

Discusión 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 A.16: Repetir no es aburrido, hay que proporcionar una identidad consistente que refuerza la sensación de “unidad” y haga que el usuario reconozca la aplicación

Relacionado con Para que los accesos a la información estén siempre visibles y consistentes, utilizar el patrón `[BP1]Navigation Bar`. Para ayudar al usuario a diferenciar entre contenidos y tipos de control (navegación y no-navegación) consultar el patrón `[MP1]Information-Interaction Decoupling`, a la vez que reconoce qué controles están asociados con qué contenidos mediante los patrones `[MP2]Information-Interaction Coupling` y `[MP3]Behavioral`

`Grouping`. Para mostrar los elementos multimedia de una manera armónica utilizar los patrones `[MP4]Define and Run Presentation` y `[MP5]Synchronise Channels`. Si la aplicación tiene condiciones de uso legales deberían colocarse siguiendo el patrón `[BP2]Footer Bar`.

Fuentes [van Duyne *et al.*, 2002]

Formalización

```
<domain:Hyperdocument rdf:ID="aplicacion"/>
<hdp:HypermediaPattern rdf:ID="AP1">
    <pattern:context rdf:resource="AN1#AN1"/>
    <pattern:context rdf:resource="A#A"/>
    <hdp:level>High</hdp:level>
    <pattern:discussion>los usuarios pueden sentirse confundidos si esperan
     encontrar el nombre de la sección en un determinado lugar y no la
      encuentran</pattern:discussion>
    <hdp:aspect>Presentation</hdp:aspect>
    <pattern:discussion>difícil la navegación por parte de los
     usuarios</pattern:discussion>
    <pattern:context rdf:resource="AE1#AE1"/>
    <pattern:name>Aesthetics</pattern:name>
    <pattern:problem>consistente e intuitiva a la vez que los nodos son mostrados
     de manera estética</pattern:problem>
    <pattern:context>la apariencia de cada una de esas unidades de información
     debe ser definida</pattern:context>
    <pattern:discussion>consistencia y la predecibilidad son atributos esenciales
     de cualquier sistema de información bien diseñado</pattern:discussion>
     <pattern:problem rdf:resource="#aplicacion"/>
     <pattern:solution rdf:resource="#aplicacion"/>
    <hdp:usesTo rdf:resource="BP1#BP1"/>
    <hdp:usesTo rdf:resource="MP1#MP1"/>
    <hdp:usesTo rdf:resource="BP2#BP2"/>
    <hdp:usesTo rdf:resource="MP2#MP2"/>
    <hdp:usesTo rdf:resource="MP4#MP4"/>
    <hdp:usesTo rdf:resource="MP3#MP3"/>
    <hdp:usesTo rdf:resource="MP5#MP5"/>
  </hdp:HypermediaPattern>
```

[BP1]Navigation Bar

Contexto Se han definido `[AN1]Multiple Ways to Navigate` y se desea hacerlos accesibles a través de toda la aplicación de manera que al usuario le resulte sencillo e intuitivo encontrarlos.

Problema ¿Cómo proporcionar acceso a las principales partes de la aplicación?

Discusión Las aplicaciones hipermedia de gran escala necesitan un esquema claro y sistemático para que los usuarios encuentren sencillo navegar por ella. Este tipo de aplicaciones suelen estar organizadas por conceptos que se centran en un determinado asunto o categoría de información.

Solución Coordinar la navegación del primer y segundo nivel de categorías en una barra de navegación a lo largo de la parte de arriba e izquierda de cada página de la aplicación. Utilizar texto con o sin iconos como enlaces dentro de la barra de navegación.

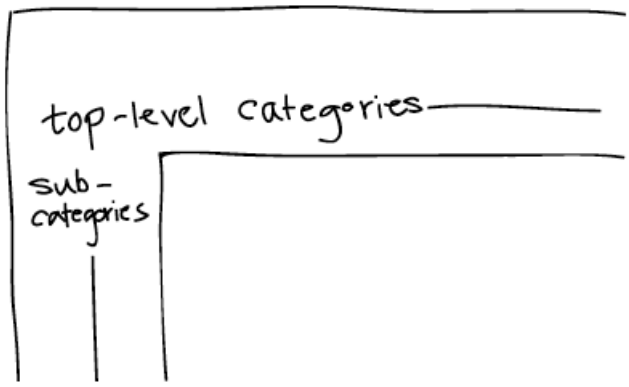


Figura A.17: Crear una barra de navegación que vaya a lo largo de la parte de arriba e izquierda de la página (imagen tomada de [van Duyne *et al.*, 2002])

Patrones relacionados Otros elementos que se pueden posicionar en la barra de navegación son los patrones `[BN2]Site Map`, `[BN3]Search Action Module` y `[BN4]One Jump Home`.

Fuentes [van Duyne *et al.*, 2002]

Formalización

```
<domain:Root rdf:ID="aplicacion">
    <domain:relationship rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >aggregation</domain:relationship>
      <domain:hasComponents>
       <domain:Node rdf:ID="barranavegacion">
         <domain:hasContents>
          <domain:Content rdf:ID="icono"/>
         </domain:hasContents>
       </domain:Node>
      </domain:hasComponents>
</domain:Root>
<domain:Anchor rdf:ID="anclas">
  <domain:location rdf:resource="#icono"/>
  </domain:Anchor>
  <domain:Link rdf:ID="indice">
  <domain:hasSource rdf:resource="#anclas"/>
  </domain:Link>
 <hdp:HypermediaPattern rdf:ID="BP1">
    <pattern:discussion> necesitan un esquema claro y sistemático para que los
     usuarios encuentren sencillo navegar por ella</pattern:discussion>
    <hdp:level>Low</hdp:level>
    <hdp:aspect>Presentation</hdp:aspect>
```

```
    <pattern:context rdf:resource="AN1#AN1"/>
    <pattern:name>Navigation Bar</pattern:name>
    <pattern:problem>acceso a las principales partes de la
     aplicación</pattern:problem>
    <pattern:problem rdf:resource="#aplicacion"/>
    <pattern:solution rdf:resource="#icono"/>
    <pattern:solution rdf:resource="#barranavegacion"/>
    <pattern:solution rdf:resource="#aplicacion"/>
    <pattern:solution rdf:resource="#indice"/>
    <hdp:suggestsTo rdf:resource="BN4#BN4"/>
    <hdp:suggestsTo rdf:resource="BN3#BN3"/>
    <hdp:suggestsTo rdf:resource="BN2#BN2"/>
    <pattern:context>hacerlos accesibles a través de toda la aplicación de manera
     que al usuario le resulte sencillo e intuitivo encontrarlos</pattern:context>
  </hdp:HypermediaPattern>
```

[MP1]Information-Interaction Decoupling

Contexto Se está diseñando la interfaz de la aplicación teniendo en cuenta [AP1]Aesthetics y se desea que el usuario sea capaz de diferenciar entre elementos de navegación y elementos de interacción.

Problema ¿Cómo diferenciar los contenidos de los diferentes tipos de control de la interfaz?

Discusión Una página de una aplicación compleja muestra diferentes contenidos, y puede estar relacionada con otras muchas páginas, proporcionando así muchas anclas. Además, si la página proporciona otras operaciones aparte de la navegación, como realizar consultas, el usuario puede experimentar una sobrecarga cognitiva. Está comprobado que cuantas más anclas se proporcionen en el texto, es más probable que el lector se distraiga y pueda no entender su significado.

La interfaz de un nodo está normalmente compuesta de elementos que muestran datos (texto o gráficos) y de elementos que proporcionan la activación de acciones, al menos de navegación. Cuando los diferentes elementos se mezclan la interacción llega a ser poco clara, y si además la información que se muestra cambia con la interacción del usuario, puede ser más difícil aún ver lo que ha cambiado.

Solución Ayudar al usuario a entender como manejar la interacción diferenciando entre la información y los elementos de interacción. Separar los canales de entrada de los canales de salida, agrupándolos en conjuntos separados. Dejar los de interacción fijos mientras que los de salida reaccionan dinámicamente a la activación del control.

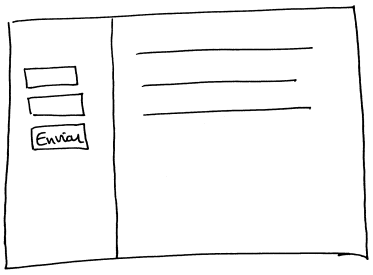


Figura A.18: Separar los controles de navegación del área de información

Patrones relacionados Una especialización de este patrón es `[MP3]Behavioral Grouping`. El opuesto a este patrón sería `[MP2]Information-Interaction Coupling`.

Fuentes [Rossi *et al.*, 2000a]

Formalización

```
<domain:Content rdf:ID="elementosinteraccion"/>
 <hdp:HypermediaPattern rdf:ID="MP1">
    <hdp:aspect>Presentation</hdp:aspect>
    <pattern:context rdf:resource="AP1#AP1"/>
    <pattern:name>Information-Interaction Decoupling</pattern:name>
    <hdp:suggestsTo rdf:resource="MP2#MP2"/>
    <pattern:discussion>el lector se distrae y puede no entender su
     significado</pattern:discussion>
    <pattern:discussion>el usuario puede experimentar una sobrecarga
     cognitiva</pattern:discussion>
    <pattern:problem>diferenciar los contenidos de los diferentes tipos de
     control</pattern:problem>
    <hdp:level>Medium</hdp:level>
    <pattern:discussion>proporciona otras operaciones aparte de la
     navegación</pattern:discussion>
    <pattern:context> sea capaz de diferenciar entre elementos de navegación y
     elementos de interacción</pattern:context>
     <pattern:solution>
      <domain:Node rdf:ID="interfaz">
        <domain:hasContents rdf:resource="#elementosinteraccion"/>
        <domain:hasContents>
          <domain:Content rdf:ID="informacion"/>
        </domain:hasContents>
      </domain:Node>
    </pattern:solution>
    <pattern:solution rdf:resource="#informacion"/>
    <pattern:solution rdf:resource="#elementosinteraccion"/>
    <pattern:problem rdf:resource="#interfaz"/>
    <hdp:specializedBy rdf:resource="MP3#MP3"/>
  </hdp:HypermediaPattern>
```

[MP2]Information-Interaction Coupling

Contexto Se está diseñando la interfaz de la aplicación teniendo en cuenta [AP1]Aesthetics y se desea que el usuario sea capaz de entender los contenidos afectados por los controles.

Problema ¿Cómo dejar claro cuál es el objeto afectado por un control en la interfaz de un nodo?

Discusión Cuando el control de la navegación u otro comportamiento es dependiente del contenido de un nodo, al separar el control de dicho contenido puede provocar desorientación al usuario.

La interfaz de un nodo está normalmente compuesta de dispositivos mostrando datos y dispositivos que proporcionan controles de activación para manipular los datos. Aspectos o datos diferentes son normalmente mostrados en la misma interfaz gráfica y se proporcionan controles para más de un aspecto utilizando para ello un nombre genérico que les resta naturalidad.

Solución Proporcionar canales de control cercanos a los datos que afectan, mediante menús o botones.

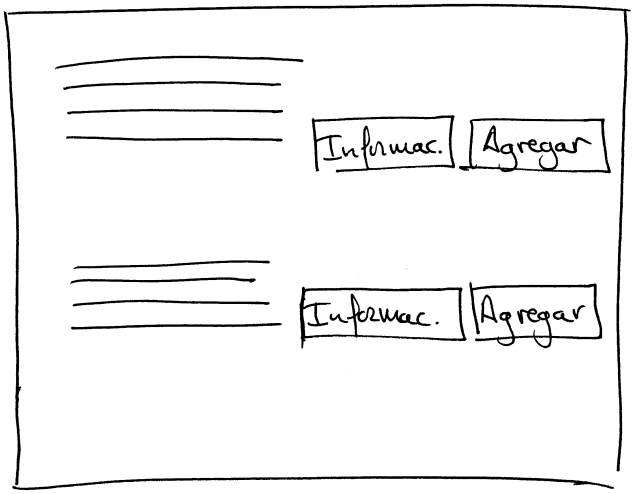


Figura A.19: Agrupar los controles junto con la información a la que afectan

Relacionado con El patrón `[MP1]Information-Interaction Decoupling` es el patrón opuesto que bajo el mismo problema, pero con diferente motivación, proporciona otra solución.

Formalización

```
<hdp:HypermediaPattern rdf:ID="MP2">
  <pattern:discussion>separar el control de dicho contenido puede provocar
   desorientación al usuario</pattern:discussion>
  <pattern:name>Information-Interaction Coupling</pattern:name>
  <hdp:aspect>Presentation</hdp:aspect>
  <pattern:context rdf:resource="AP1#AP1"/>
  <hdp:suggestsTo rdf:resource="MP1#MP1"/>
  <pattern:context>el usuario sea capaz de entender los contenidos afectados por
   los controles</pattern:context>
  <pattern:discussion>el control de la navegación u otro comportamiento es
   dependiente del contenido de un nodo</pattern:discussion>
  <pattern:problem>el objeto afectado por un control en la interfaz de un
   nodo</pattern:problem>
   <pattern:solution>
    <domain:Node rdf:ID="interfaz">
      <domain:hasContents>
        <domain:Content rdf:ID="botones"/>
      </domain:hasContents>
      <domain:hasContents>
        <domain:Content rdf:ID="datos"/>
      </domain:hasContents>
    </domain:Node>
  </pattern:solution>
  <pattern:solution rdf:resource="#datos"/>
  <pattern:solution rdf:resource="#botones"/>
  <pattern:problem rdf:resource="#interfaz"/>
  <hdp:level>Medium</hdp:level>
</hdp:HypermediaPattern>
```

[MP3]Behavioral Grouping

Contexto Se han separado los elementos de interacción de aquellos que muestran información mediante `[MP1]Information-Interaction Decoupling`, pero puede haber diferentes grupos de elementos de interacción.

Problema ¿Cómo organizar los diferentes tipos de controles en la interfaz para que los usuarios puedan entenderlos fácilmente?

Discusión Un problema habitual cuando se está construyendo la interfaz de una aplicación hipermedia es cómo organizar los elementos de control, como anclas, botones, etc., para proporcionar una interfaz significativa.

En una aplicación típica, hay diferentes tipos de elementos activos en la interfaz, como los que proporcionan funcionalidades de navegación o funcionalidades de la

aplicación. Incluso si se decide separar los contenidos de información de los controles de interacción puede que haya diferentes tipos de actividades de interacción que necesitemos organizar.

Solución Agrupar los objetos de control según su funcionalidad (global, contextual, estructural y aplicación) y hacerlos visibles en diferentes áreas de la pantalla. Los grupos típicos de las aplicaciones hipermedia pueden ser de: navegación global (vuelta atrás, contenidos, histórico, etc), anclas para nodos relacionados("véase" y otras relaciones con más semántica), controles de interfaz (botones implementando `[MI1]Information on Demand` por ejemplo), otras funcionalidades no relacionadas con la hipermedia directamente (enviar un formulario).

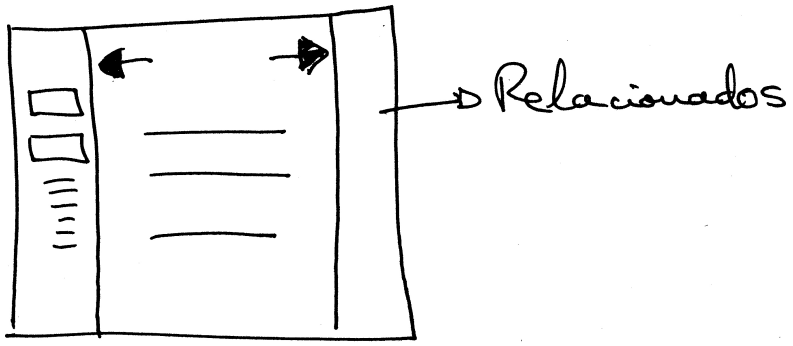


Figura A.20: Agrupar los controles según su funcionalidad

Relacionado con

Fuentes [Rossi *et al.*, 2000a]

Formalización

```
<hdp:HypermediaPattern rdf:ID="MP3">
   <pattern:name>Behavioral Grouping</pattern:name>
   <pattern:discussion>hay diferentes tipos de elementos activos en la
    interfaz</pattern:discussion>
   <hdp:level>Medium</hdp:level>
   <pattern:problem> organizar los diferentes tipos de controles en la interfaz
    para que los usuarios puedan entenderlos fácilmente</pattern:problem>
   <pattern:discussion>para proporcionar una interfaz
    significativa</pattern:discussion>
   <hdp:aspect>Presentation</hdp:aspect>
   <pattern:context>puede haber diferentes grupos de elementos de
    interacción</pattern:context>
   <pattern:context rdf:resource="MP1#MP1"/>
   <pattern:solution>
```

```
      <domain:Node rdf:ID="interfaz">
        <domain:hasContents>
          <domain:Content rdf:ID="navegacionglobal"/>
        </domain:hasContents>
        <domain:hasContents>
          <domain:Content rdf:ID="nodosrelacionados"/>
        </domain:hasContents>
        <domain:hasContents>
          <domain:Content rdf:ID="controles"/>
        </domain:hasContents>
      </domain:Node>
    </pattern:solution>
    <pattern:solution rdf:resource="#controles"/>
    <pattern:solution rdf:resource="#nodosrelacionados"/>
    <pattern:solution rdf:resource="#navegacionglobal"/>
    <pattern:problem rdf:resource="#interfaz"/>
  </hdp:HypermediaPattern>
```

[MP4]Define and Run Presentation

Contexto Se está diseñando la interfaz de la aplicación teniendo en cuenta [AP1]Aesthetics y se desea que una colección organizada de componentes sea mostrada en secuencia al usuario.

Problema ¿Cómo hacer para que el usuario perciba los elementos multimedia simultáneamente?

Discusión Todos los componentes están organizados de acuerdo a un orden parcial, p.e. jerárquico o en red. A veces un conjunto de elementos multimedia deben presentarse al usuario de forma secuencial o simultánea. Por ejemplo, una serie de contenidos, donde los gráficos son acompañados con sonidos. Estos contenidos suelen ser asíncronos.

Solución Planear la presentación de los contenidos multimedia organizándolos en un orden parcial. Posicionarlos en una línea de tiempo para elegir cuando se muestra cada elemento.

Relacionado con Este patrón puede emplearse junto con [MP5]Synchronise Channels para mostrar una presentación de componentes multimedia sincronizados.

Fuentes [Cybulski y Linden, 1999]

Formalización

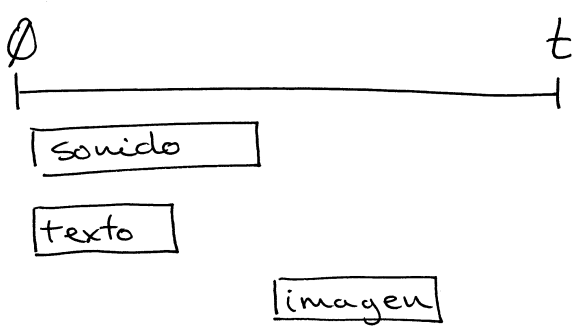


Figura A.21: Organizar los contenidos multimedia en una línea del tiempo según su aparición

```
<hdp:HypermediaPattern rdf:ID="MP4">
   <pattern:name>Define and Run Presentation</pattern:name>
   <hdp:level>Medium</hdp:level>
   <pattern:context rdf:resource="AP1#AP1"/>
   <pattern:discussion>Estos contenidos suelen ser asíncronos</pattern:discussion>
   <pattern:problem>el usuario perciba los elementos multimedia
    simultáneamente</pattern:problem>
   <pattern:context>una colección organizada de componentes sea mostrada en
    secuencia al usuario</pattern:context>
   <hdp:aspect>Presentation</hdp:aspect>
   <pattern:discussion>un conjunto de elementos multimedia deben presentarse al
    usuario de forma secuencial o simultánea</pattern:discussion>
   <pattern:solution rdf:resource="#elementosmultimedia"/>
   <pattern:problem rdf:resource="#elementosmultimedia"/>
   <pattern:solution>
     <domain:Synchronization rdf:ID="orden"/>
   </pattern:solution>
   <hdp:suggestsTo rdf:resource="MP5#MP5"/>
 </hdp:HypermediaPattern>
```

[MP5]Synchronise Channels

Contexto Se está diseñando la interfaz de la aplicación teniendo en cuenta [AP1]Aesthetics y se desea que el usuario perciba los elementos multimedia como una composición armónica.

Problema ¿Cómo sincronizar un conjunto de elementos continuos (como vídeo o sonido) y no continuos (como texto o imágenes)?

Discusión Contenidos continuos y no continuos necesitan ser sincronizados en una presentación. Por ejemplo, sincronización de voz con texto o con gráficos para definir

una presentación narrativa. Los contenidos continuos tienen propiedades temporales, mientras que los no continuos carecen de ellas. Sin embargo la inclusión de la dimensión temporal incrementa la complejidad en la planificación y difusión de la presentación.

Solución Para los elementos no continuos, definir elementos de retardo. Los retardos definen la secuencia temporal de posiciones dentro de un conjunto de elementos continuos y no continuos.

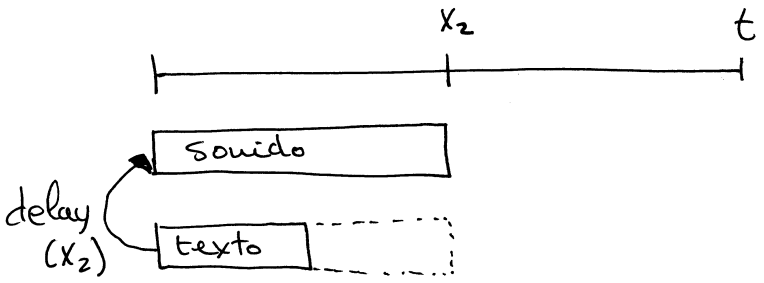


Figura A.22: Definir retardos para sincronizar elementos no continuos con elementos continuos

Relacionado con Para organizar los contenidos multimedia según un orden de aparición véase el patrón `[MP4]Define And Run Presentation`.

Fuentes [Cybulski y Linden, 1999]

Formalización

```
<hdp:HypermediaPattern rdf:ID="MP5">
    <pattern:discussion>Contenidos continuos y no continuos necesitan ser
     sincronizados en una presentación</pattern:discussion>
    <pattern:name>Synchronise Channels</pattern:name>
    <pattern:problem>sincronizar un conjunto de elementos
     continuos</pattern:problem>
    <pattern:context>el usuario perciba los elementos multimedia como una
     composición armónica</pattern:context>
    <pattern:problem>y no continuos</pattern:problem>
    <pattern:context rdf:resource="AP1#AP1"/>
    <hdp:suggestsTo rdf:resource="MP4#MP4"/>
    <hdp:level>Medium</hdp:level>
    <hdp:aspect>Presentation</hdp:aspect>
    <pattern:problem>
      <domain:Content rdf:ID="elementosnocontinuos"/>
      <domain:Content rdf:ID="elementoscontinuos"/>
    </pattern:problem>
    <pattern:problem rdf:resource="#elementoscontinuos"/>
    <pattern:solution rdf:resource="#elementoscontinuos"/>
    <pattern:solution rdf:resource="#elementosnocontinuos"/>
```

```
    <pattern:solution>
      <domain:Synchronization rdf:ID="retardo">
        <domain:hasTarget>
        <domain:Anchor rdf:ID="siguiente">
            <domain:location rdf:resource="#elementoscontinuos"/>
          </domain:Anchor>
        </domain:hasTarget>
        <domain:hasSource>
          <domain:Anchor rdf:ID="anclas">
            <domain:location rdf:resource="#elementosnocontinuos"/>
          </domain:Anchor>
        </domain:hasSource>
      </domain:Synchronization>
    </pattern:solution>
  </hdp:HypermediaPattern>
```

[BP2]Footer Bar

Contexto Se está definiendo la composición de los elementos de información de los nodos mediante [AP1]Aesthetics y se desea poner algún tipo de información al final del espacio de visualización del nodo.

Problema ¿Qué elementos deberían aparecer al final del espacio de visualización de un nodo?

Discusión La aplicación puede contener material que está protegido por derechos de autor o contiene datos personales de los usuarios.

Solución Añadir al final de la página una barra con enlaces a las condiciones de uso de la aplicación o diferentes usos legales. Hacerlos accesibles a través de todas las páginas de la aplicación. Esta barra también es apropiada para repetir accesos a los principales nodos de información u otros tipos de navegación que puedan venir bien.

Relacionado con

Fuentes [33]

Formalización

```
<hdp:HypermediaPattern rdf:ID="BP2">
    <hdp:level>Low</hdp:level>
    <pattern:context rdf:resource="AP1#AP1"/>
    <pattern:problem> elementos deberían aparecer al final del espacio de
     visualización de un nodo</pattern:problem>
    <pattern:discussion>material que está protegido por derechos de autor o
     contiene datos personales de los usuarios</pattern:discussion>
    <pattern:context>poner algún tipo de información al final del espacio de
     visualización del nodo</pattern:context>
```

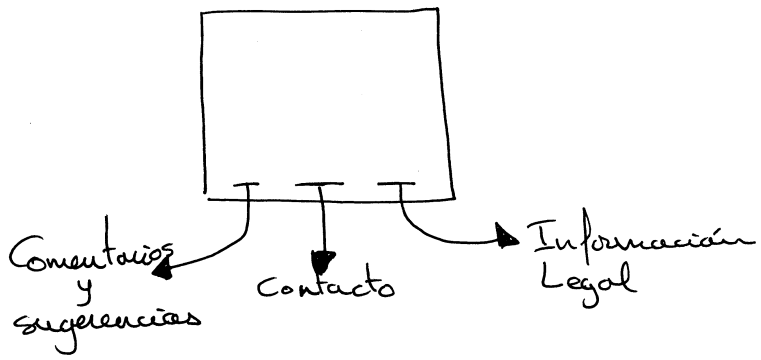


Figura A.23: El usuario debe conocer en todo momento las condiciones legales bajo las que está sujeta la aplicación

```
  <pattern:name>Footer Bar</pattern:name>
  <hdp:aspect>Presentation</hdp:aspect>
  <pattern:solution rdf:resource="#barraenlaces"/>
  <pattern:problem>
   <domain:Content rdf:ID="barraenlaces"/>
   <domain:Node rdf:ID="nodo">
      <domain:hasContents rdf:resource="#barraenlaces"/>
    </domain:Node>
  </pattern:problem>
  <pattern:solution>
    <domain:Node rdf:ID="condiciones"/>
  </pattern:solution>
  <pattern:solution rdf:resource="#nodo"/>
</hdp:HypermediaPattern>
```

A.1.4. [AI1]Interaction

Una aplicación hipermedia es un sistema interactivo, por lo cual es necesario proporcionar al usuario mecanismos que le permitan interaccionar con la aplicación, tanto para explorar como para manipular los datos, siendo consciente del alcance de sus acciones. El conjunto de patrones recogidos bajo el nombre `Interaction` pretende mejorar la comunicación entre el usuario y la aplicación.

[AI1]Interaction

Contexto Se está diseñando una `[A]Hypermedia Application` y uno de los modos de interacción es la navegación definida en `[AN1]Multiple Ways to Navigate`, pero es necesario mejorar la comunicación entre el usuario y la aplicación.

Problema ¿Cómo hacer que el usuario sienta que conoce y controla el proceso de interacción con respecto a la información y a las tareas que proporciona la aplicación?

Discusión 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 a las acciones de los usuarios [Ebersole, 1997]. Al principio las aplicaciones hipermedia eran sólo utilizadas para proporcionar información. Sin embargo, el uso de la web ha hecho que se expandan 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 quien 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.

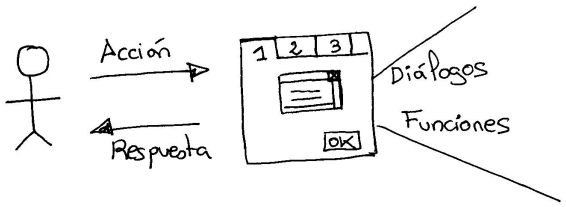


Figura A.24: Informar al usuario sobre el efecto de sus acciones permitiéndole evaluar su objetivo final a alcanzar y determinar el siguiente paso para conseguirlo

Patrones relacionados Para que el usuario sea el que controle cuánta información desea visualizar en un nodo utilizar el patrón `[MI1]Information on Demand`. Para informar al usuario del estado de la interacción de tal modo que sepa qué esperar consultar el patrón `[MI2]Process Feed-back`. Aquellos elementos cuya activación provoquen la ejecución de una acción deberían ser presentados como `[BI1]Action Buttons`.

Fuentes

Formalización

```
<domain:Hyperdocument rdf:ID="aplicacion"/>
 <hdp:HypermediaPattern rdf:ID="AI1">
   <pattern:name>Interaction</pattern:name>
   <hdp:aspect>Interaction</hdp:aspect>
   <pattern:discussion>sino se impacientará y cometerá
    errores</pattern:discussion>
   <hdp:usesTo rdf:resource="MI1#MI1"/>
   <pattern:context>mejorar la comunicación entre el usuario y la
    aplicación</pattern:context>
   <pattern:problem> el usuario sienta que conoce y controla el proceso de
    interacción</pattern:problem>
   <hdp:level>High</hdp:level>
   <pattern:discussion>interacción implica un diálogo entre dos
    partes</pattern:discussion>
   <pattern:context rdf:resource="AN1#AN1"/>
   <pattern:discussion>es él quien controla la marcha de los
    procesos</pattern:discussion>
   <hdp:usesTo rdf:resource="MI2#MI2"/>
   <pattern:context rdf:resource="A#A"/>
   <pattern:problem rdf:resource="#aplicacion"/>
   <hdp:usesTo rdf:resource=" BI1# BI1"/>
 </hdp:HypermediaPattern>
```

[MI1]Information on Demand

Contexto La comunicación entre el usuario y la aplicación se puede mejorar con [AI1]Interaction, en concreto, se desea que el usuario pueda controlar la cantidad de información que recibe.

Problema ¿Cómo organizar la interfaz de un nodo de tal manera que sea el usuario el que decida cúanta información extra desea recibir?

Discusión Normalmente es difícil decidir cómo mostrar los contenidos y anclas en un nodo. Desafortunadamente, la pantalla es más pequeña de lo que realmente se necesita y muchas veces no se puede hacer uso de otros medios como mostrar una imagen a la vez que suena un sonido, bien por razones tecnológicas o cognitivas. Por ejemplo, en una aplicación de pinturas se quiere mostrar diferentes detalles de un cuadro, como el nombre del pintor, el año, el museo, descripción técnica, etc., pero esto es difícil si se quiere maximizar el espacio dedicado a la imagen del cuadro.

Un problema común aparece cuando un nodo tiene una gran cantidad de información y no hay espacio suficiente en una pantalla para mostrarlo todo junto, o hay elementos multimedia que pueden distraer al usuario, como por ejemplo un sonido. Además, la

barra de desplazamiento (*scroll*) puede no ser aceptable porque el lector no consigue una vista total sobre lo que él encontrará en el nodo.

Finalmente, se puede optar por particionar el nodo en diferentes ventanas y representar así toda la información, definiendo enlaces entre estos nuevos nodos. Esto también es un problema porque en el intento de corresponder el diseño con un asunto de implementación, se ha corrompido la estructura global de navegación de la aplicación. El usuario puede desorientarse al tener la impresión de tratar con diferentes entidades, cuando de hecho está accediendo a otra parte de la misma entidad conceptual.

Solución Presentar solamente un subconjunto de atributos (los más importantes) y dejar al usuario que controle qué información adicional necesita en la pantalla mediante objetos activos de la interfaz (botones).

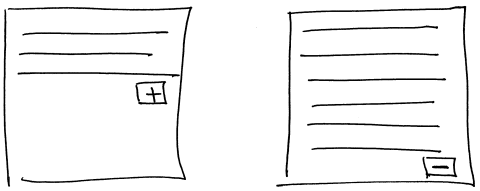


Figura A.25: Añadir botones o controles activos que regulen la información presentada al usuario

La activación de estos botones no producen navegación, sino que diferentes atributos del mismo nodo sean mostrados.

Relacionados Para que el usuario sea capaz de identificar el control de la interacción, se puede utilizar el patrón `[BI1]Action Buttons`.

Para determinar qué cantidad de información debería contener un nodo, se puede consultar el patrón `[BE2]Node as a single unit`.

Para señalar al usuario el objeto afectado por la activiación de un control, se puede consultar el patrón `[MP2]Infomation-Interaction Coupling`.

Fuentes [Garrido *et al.*, 1997]

Formalización

```
<domain:Content rdf:ID="atributos"/>
<domain:Content rdf:ID="boton">
   <hasEvents>
```

```
      <domain:Event rdf:ID="masinformacion"/>
     </hasEvents>
   </domain:Content>
<hdp:HypermediaPattern rdf:ID="MI1">
    <hdp:level>Medium</hdp:level>
    <pattern:name>Information on Demand</pattern:name>
    <pattern:discussion>un nodo tiene una gran cantidad de
     información</pattern:discussion>
    <pattern:context rdf:resource="AI1#AI1"/>
    <pattern:discussion> la pantalla es más pequeña de lo que realmente se
     necesita</pattern:discussion>
    <hdp:suggestsTo rdf:resource="BE2#BE2"/>
    <pattern:problem>el usuario el que decida cúanta información extra desea
     recibir</pattern:problem>
    <hdp:aspect>Interaction</hdp:aspect>
    <hdp:suggestsTo rdf:resource="MP2#MP2"/>
    <pattern:context>el usuario pueda controlar la cantidad de información que
     recibe</pattern:context>
    <pattern:discussion>es difícil decidir cómo mostrar los contenidos y anclas en
     un nodo</pattern:discussion>
    <pattern:discussion>se ha corrompido la estructura global de navegación de la
     aplicación</pattern:discussion>
    <pattern:discussion>por particionar el nodo en diferentes
     ventanas</pattern:discussion>
     <pattern:solution rdf:resource="#boton"/>
    <pattern:problem>
      <domain:Node rdf:ID="nodo">
        <domain:hasContents rdf:resource="#atributos"/>
      </domain:Node>
    </pattern:problem>
    <pattern:solution rdf:resource="#atributos"/>
    <pattern:solution rdf:resource="#nodo"/>
  </hdp:HypermediaPattern>
```

[MI2]Process Feed-back

Contexto Se está mejorando la comunicación entre el usuario y la aplicación con `[AI1]Interaction` y se desea que cuando el sistema esté realizando algún tipo de operación se mantenga informado al usuario.

Problema ¿Cómo mantener al usuario informado sobre el estado de la interacción de tal modo que sepa qué esperar?

Discusión Cuando el usuario interactúa con una aplicación hipermedia, puede suceder que muchas de las operaciones resulten ser operaciones no-atómicas, como conseguir información de una base de datos, cargar una imagen o contactar con otra máquina. En estos casos, el usuario puede sentir que el sistema no recibió la orden o simplemente no está funcionando de manera correcta. El usuario puede perder su paciencia al no conseguir una respuesta positiva del sistema, comenzado a reintentar la última acción o incluso realizar otras con la intención de conseguir alguna respuesta. En este punto,

el usuario puede haber ya perdido la orientación y la tarea que se estaba realizando se ha dejado en un segundo plano.

Solución Analizar qué operaciones son atómicas y cuáles necesitan ser controladas. Para operaciones no-atómicas, dar información sobre el actual estatus: comienzo, progreso y finalización de la operación. El tipo de información depende del tipo de operación involucrada: conexión de red, carga de fichero o búsqueda en base de datos.

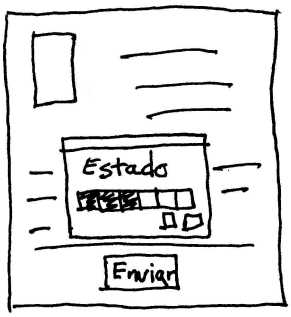


Figura A.26: Si la operación a realizar llevará mucho tiempo, mostrar como va progresando

Relacionado con Utilizar el patrón `[BI1]Action Buttons` para activar el proceso.

Fuentes [Garrido *et al.*, 1997]

Formalización

```
<domain:Content rdf:ID="informacion"/>
<hdp:HypermediaPattern rdf:ID="MI2">
   <hdp:aspect>Interaction</hdp:aspect>
   <pattern:context>cuando el sistema esté realizando algún tipo de operación se
    mantenga informado al usuario</pattern:context>
   <pattern:name>Process Feed-back</pattern:name>
   <pattern:context rdf:resource="AI1#AI1"/>
   <pattern:discussion>El usuario puede perder su paciencia al no conseguir una
    respuesta positiva</pattern:discussion>
   <pattern:problem> el estado de la interacción de tal modo que sepa qué
    esperar</pattern:problem>
   <pattern:discussion>operaciones resulten ser operaciones
    no-atómicas</pattern:discussion>
   <hdp:level>Medium</hdp:level>
    <pattern:problem>
     <domain:Node rdf:ID="estadointeraccion">
       <domain:hasContents rdf:resource="#informacion"/>
     </domain:Node>
   </pattern:problem>
   <pattern:solution rdf:resource="#informacion"/>
   <pattern:solution rdf:resource="#estadointeraccion"/>
 </hdp:HypermediaPattern>
```

Figura A.27: Un botón representando una acción (imagen tomada de [van Duyne *et al.*, 2002])

[BI1]Action Buttons

Contexto Se está mejorando la comunicación entre el usuario y la aplicación con `[AI1]Interaction`. Hay algunos enlaces que representan acciones y no navegación, como en el caso de `[MI1]Information on Demand`, y se desea que el usuario sea consciente de la importancia que conlleva dicha acción.

Problema ¿Cómo diferenciar los enlaces corrientes de los enlaces que representan acciones?

Discusión Los enlaces de texto son buenos para moverse de nodo a nodo, el usuario sabe que no hay efectos laterales o consecuencias, pero no son bastante correctos para representar acciones que hacen algo importante, como autorizar una compra o enviar un mensaje a un tablón de mensajes.

Solución Usar botones para representar acciones. Los botones son más convenientes que los enlaces textuales porque en el mundo físico causan acciones, como el pulsar el botón de un mando causa que la televisión cambie de canal o pulsar el botón de un ascensor hace ir a una determinada planta.

Si se utilizan imágenes, hacer que parezcan que ellas pueden ser pinchadas dándolas una apariencia tridimensional. También proporcionar etiquetas concisas para explicar lo que los botones harán.

Relacionado con `Action Buttons` son utilizados para llevar a cabo acciones no atómicas, utilizar `[MI2]Process Feed-back` para mostrar el estado de la acción en curso.

Fuentes [van Duyne *et al.*, 2002],[33]

Formalización

```
<domain:Content rdf:ID="boton">
    <hasEvents>
      <domain:Event rdf:ID="acciones"/>
    </hasEvents>
  </domain:Content>
<hdp:HypermediaPattern rdf:about="# BI1">
    <hdp:suggestsTo rdf:resource="MI2#MI2"/>
    <pattern:name>Action Buttons</pattern:name>
    <pattern:problem>diferenciar los enlaces corrientes de los enlaces que
     representan acciones</pattern:problem>
    <hdp:level>Low</hdp:level>
    <pattern:context rdf:resource="MI1#MI1"/>
    <hdp:aspect>Interaction</hdp:aspect>
    <pattern:context rdf:resource="AI1#AI1"/>
    <pattern:context>el usuario sea consciente de la importancia que conlleva
     dicha acción</pattern:context>
     <pattern:solution rdf:resource="#boton"/>
    <pattern:problem rdf:resource="#acciones"/>
  </hdp:HypermediaPattern>
```

A.1.5. [AZ1]Personalization

Muchas aplicaciones web tienen que soportar diferentes roles de usuarios, incluyendo el de administrador de la aplicación. Estos usuarios tienen diferentes vistas de la aplicación y requieren la construcción de aplicaciones web personalizadas las cuales son sensibles a las necesidades individuales de cada usuario o grupo de usuarios. El conjunto de patrones aquí recogidos bajo la categoría `Personalization` guían al diseñador en la construcción modular y evolutiva de una aplicación web personalizada, separando la estructura de la información de la navegación y de la interfaz.

`[AZ1]Personalization`

Contexto Se está diseñando una `[A]Hypermedia Application` donde se ha definido una `[AE1]User-centered Structure`, diferentes tipos de acceso con `[AN1]Multiple Ways to Navigate`, el aspecto estético y cognitivo de los contenidos con `[AP1]Aesthetics` y cómo responder a la interacción con el usuario en `[AI1]Interaction`, 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 hacer para que el usuario pueda acceder a una aplicación adaptada a sus intereses y necesidades?

Discusión 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.

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.

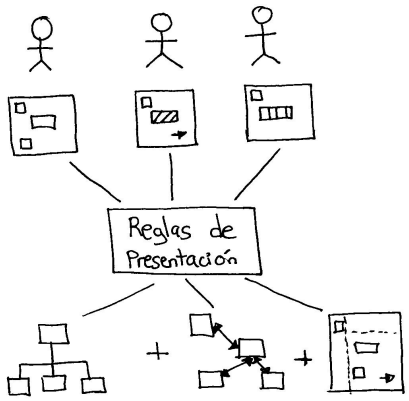


Figura A.28: Separar la estructura de la información de los contenidos para que cada usuario obtenga una vista diferente sin que el mantenimiento de la aplicación se vea penalizado

Relacionado Para definir los diferentes tipos de usuarios de la aplicación utilizar `[MZ1]Role Subtype`. Si se quiere restringir el espacio de visualización a los intereses de cada tipo de usuario consultar `[MZ2]Structure Personalization`. En el caso de querer personalizar los valores de los contenidos usar `[MZ3]Content Personalization`.

El patrón `[MS1]Authorization` puede ser adaptado para definir reglas de personalización, cambiando las categorías de acceso a nodos y contenidos por especificaciones de presentación [Aedo *et al.*, 2002b].

Fuentes [Rossi *et al.*, 2001b],[Aedo *et al.*, 2002b]

Formalización

```
<hdp:HypermediaPattern rdf:ID="AZ1">
   <pattern:problem>
     <domain:Hyperdocument rdf:ID="aplicación"/>
   </pattern:problem>
   <pattern:problem>acceder a una aplicación adaptada a sus intereses y
    necesidades</pattern:problem>
   <pattern:context rdf:resource="AI1#AI1"/>
   <hdp:usesTo rdf:resource="MZ3 #MZ3 "/>
   <pattern:discussion>reutilizar aquellos aspectos que son
    comunes</pattern:discussion>
   <hdp:level>High</hdp:level>
   <hdp:suggestsTo rdf:resource="MS1#MS1"/>
   <pattern:discussion>no se trata de construir aspectos
    específicos</pattern:discussion>
   <pattern:discussion>adaptar la información o servicios
    proporcionados</pattern:discussion>
   <pattern:context rdf:resource="AE1#AE1"/>
   <pattern:name>Personalization</pattern:name>
   <hdp:usesTo rdf:resource="MZ1#MZ1"/>
   <hdp:usesTo rdf:resource="MZ2#MZ2"/>
   <hdp:aspect>Personalization</hdp:aspect>
   <pattern:context rdf:resource="A#A"/>
   <pattern:discussion> las necesidades de cada usuario específico o conjunto de
    usuarios</pattern:discussion>
   <pattern:context rdf:resource="AP1#AP1"/>
   <pattern:context>adaptar la aplicación a las necesidades de cada
    usuario</pattern:context>
   <pattern:discussion>la navegación de los usuarios una tarea
    dura</pattern:discussion>
   <pattern:context rdf:resource=" AN1# AN1"/>
    <pattern:solution>
     <domain:User rdf:about="#usuarios"/>
   </pattern:solution>
 </hdp:HypermediaPattern>
```

[MZ1]Role Subtype

Contexto Se está personalizando la aplicación mediante `[AZ1]Personalization` y la cuestión más importante es identificar los usuarios.

Problema ¿Cómo representar a los diferentes usuarios de la aplicación sin tener que modelar a cada uno de ellos?

Discusión El perfil del usuario es un componente importante en las aplicaciones personalizadas. Sin embargo, cuando la cantidad de usuarios de un sistema es alta, es muy duro personalizar la aplicación para cada uno de ellos. En realidad, se pueden distinguir grupos de usuarios que suelen comportarse de forma similar, es decir, que comparten

una serie de intereses o de tareas. Por ejemplo, en una aplicación de revisión de artículos se pueden identificar tres tipos de roles: autores, revisores, y comité de programa. Cada usuario está interesado en diferentes categorías de información. Los autores visitan la aplicación para explorar información de la conferencia y enviar artículos; los revisores descargan artículos y envían sus revisiones; el comité de programa debería poder descargar información sobre el aviso de artículos, tutoriales o *workshops* y revisar recomendaciones sobre cada artículo.

Solución Agrupar a los diferentes tipos de usuarios por similitudes. Modelar cada tipo de usuario como un rol, los cuales representan trabajos o responsabilidades. Definir una jerarquía de roles como un grafo dirigido acíclico, donde los roles generales son especializados en roles más concretos que heredan las propiedades y reglas de sus padres.

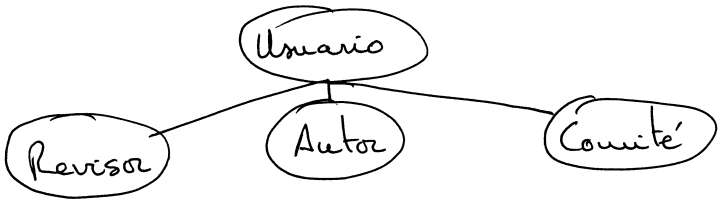


Figura A.29: Agrupar los usuarios en roles alivia las tareas de administración

Relacionado con Si se quiere restringir el espacio de visualización a los intereses de cada rol consultar `[MZ2]Structure Personalization`. En el caso de querer personalizar los valores de los contenidos usar `[MZ3]Content Personalization`.

Fuentes [Fowler, 1997b]

Formalización

```
<hdp:HypermediaPattern rdf:ID="MZ1">
   <pattern:discussion>perfil del usuario es un componente
    importante</pattern:discussion>
   <pattern:context rdf:resource="AZ1#AZ1"/>
   <pattern:name>Role Subtype</pattern:name>
   <pattern:problem>representar a los diferentes usuarios</pattern:problem>
   <hdp:level>Medium</hdp:level>
   <pattern:discussion>usuarios que suelen comportarse de forma
    similar</pattern:discussion>
   <hdp:suggestsTo rdf:resource="MZ2#MZ2"/>
   <hdp:suggestsTo rdf:resource="MZ3#MZ3"/>
   <hdp:aspect>Personalization</hdp:aspect>
```

```
  <pattern:discussion> cuando la cantidad de usuarios de un sistema es alta, es
   muy duro personalizar la aplicación para cada uno de
    ellos</pattern:discussion>
  <pattern:context>identificar los usuarios</pattern:context>
  <pattern:solution>
    <domain:User rdf:ID="generales">
      <domain:userList>
        <domain:User rdf:ID="concretos"/>
      </domain:userList>
      <domain:userType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >role</domain:userType>
    </domain:User>
  </pattern:solution>
  <pattern:problem rdf:resource="#generales"/>
</hdp:HypermediaPattern>
```

[MZ2] Structure Personalization

Contexto Se está personalizando la aplicación mediante `[AZ1]Personalization` y se han identificado ya a los usuarios en `[MZ1] Role Subtype`. Ahora se desea que cada tipo de usuario vea sólo las secciones y detalles sobre las que está interesado.

Problema ¿Cómo mostrar a cada tipo de usuario el espacio de información que le interesa sin duplicar la estructura?

Discusión Ciertos tipos de aplicaciones, como los portales, no sólo involucran una gran cantidad de información sino también una gran variedad de temas y servicios. Cuando el usuario se enfrenta a este tipo de aplicaciones puede encontrarse agobiado no sólo por el número de posibles enlaces a seguir, sino por la diversidad de temas y servicios. Una solución a este problema es organizar los temas mediante una taxonomía, como recoge el patrón `[ME1]Hierarchical Organization`, pero el resultado puede ser que el usuario sea reacio a navegar a las categorías de niveles más bajos, así perdiendo la oportunidad de alcanzarlos.

Los usuarios pueden no llegar a encontrar lo que quieren, cuando el espacio de información es demasiado denso. Incluso cuando encuentran lo que estaban buscando, les gustaría haberlo hecho de una manera más sencilla.

Además, diferentes usuarios pueden estar interesados en diferentes temas o algunos pueden no estarlo en un determinado tema o servicio.

Solución Personalizar la estructura de la aplicación decidiendo para cada rol el acceso a aquellos nodos y contenidos de información que le interesan. Así se define una

vista de la estructura de la información diferente para cada rol a la vez que se evita duplicarla. En cuanto a la navegación, si alguno de los nodos no seleccionados para formar parte de la vista del usuario son origen o destino de un enlace, dicho enlace y sus anclas no aparecerán.

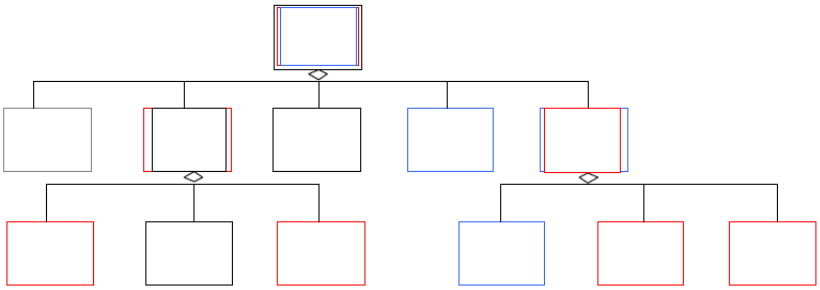


Figura A.30: Determinar para cada usuario los nodos y los contenidos que serán accesibles (imagen tomada de [Rossi *et al.*, 2001a])

Relacionado con Para determinar la información que presentará cada contenido según el usuario utilizar `[MZ3]Content Personalization`. Adaptar el patrón `[MS1]Authorization` para definir reglas de personalización, cambiando las categorías de acceso a nodos y contenidos por especificaciones de presentación.

Fuentes [Rossi *et al.*, 2001a]

Formalización

```
<hdp:HypermediaPattern rdf:ID="MZ2">
    <hdp:suggestsTo rdf:resource="MS1#MS1"/>
    <pattern:name>Structure Personalization</pattern:name>
    <pattern:discussion>enlaces a seguir</pattern:discussion>
    <pattern:context>cada tipo de usuario vea sólo las secciones y detalles sobre
     las que está interesado</pattern:context>
    <hdp:level>Low</hdp:level>
    <pattern:discussion>el usuario se enfrenta a este tipo de aplicaciones puede
     encontrarse agobiado</pattern:discussion>
    <pattern:discussion>diferentes usuarios pueden estar interesados en diferentes
     temas </pattern:discussion>
    <pattern:problem>mostrar a cada tipo de usuario el espacio de información que
     le interesa sin duplicar la estructura</pattern:problem>
    <pattern:context rdf:resource="MZ1#MZ1"/>
    <pattern:discussion>usuarios pueden no llegar a encontrar lo que
     quieren</pattern:discussion>
    <pattern:context rdf:resource="AZ1#AZ1"/>
    <hdp:suggestsTo rdf:resource="MZ3#MZ3"/>
    <hdp:aspect>Personalization</hdp:aspect>
    <pattern:discussion>diversidad de temas</pattern:discussion>
    <pattern:solution>
      <domain:AccessCategory rdf:ID="acceso">
```

```
      <domain:hasUser>
        <domain:User rdf:ID="usuario"/>
      </domain:hasUser>
      <domain:hasObject>
        <domain:Node rdf:ID="nodos"/>
      </domain:hasObject>
      <domain:securityCategory rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Browsing</domain:securityCategory>
    </domain:AccessCategory>
  </pattern:solution>
  <pattern:solution rdf:resource="#nodos"/>
  <pattern:problem rdf:resource="#usuario"/>
  <pattern:solution rdf:resource="#usuario"/>
</hdp:HypermediaPattern>
```

[MZ3]Content Personalization

Contexto Se está personalizando la aplicación mediante `[AZ1]Personalization` y se desea proporcionar a los usuarios datos personalizados.

Problema ¿Cómo hacer que el valor de los contenidos sean personalizados?

Discusión En muchas aplicaciones web se proporciona a los usuarios contenidos ligeramente diferentes sobre elementos concretos de información. Por ejemplo, diferentes compradores en una tienda virtual pueden pagar diferentes precios según su historial de compra, o mostrar el precio en su moneda preferida.

Solución Permitir que el valor de los contenidos de los nodos varíe según el rol del usuario. Esto significa que el valor de un contenido debería ser tratado de manera independiente al propio contenido y unido mediante una función que permita asignar al contenido de un determinado valor según el tipo de usuario.

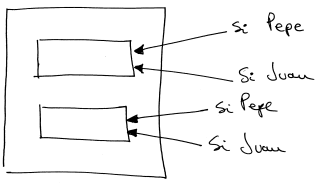


Figura A.31: Hacer que los valores de los contenidos dependan de cada usuario mediante una función de asignación

Relacionado con Adaptar el patrón `[MS1]Authorization` para definir reglas de personalización, cambiando las categorías de acceso a los contenidos por funciones de presentación [Montero *et al.*, 2002].

Fuentes [Rossi *et al.*, 2001a]

Formalización

```
<hdp:HypermediaPattern rdf:about="#MZ3 ">
    <pattern:discussion>proporciona a los usuarios contenidos ligeramente
     diferentes sobre elementos concretos</pattern:discussion>
    <pattern:problem>el valor de los contenidos sean
     personalizados</pattern:problem>
    <pattern:context rdf:resource="AZ1#AZ1"/>
    <hdp:suggestsTo rdf:resource="MS1#MS1"/>
    <hdp:level>Medium</hdp:level>
    <pattern:context>proporcionar a los usuarios datos
     personalizados</pattern:context>
    <hdp:aspect>Personalization</hdp:aspect>
    <pattern:name>Content Personalization</pattern:name>
    <pattern:solution>
      <domain:AccessCategory rdf:ID="acceso">
        <domain:hasUser>
          <domain:User rdf:ID="usuario"/>
        </domain:hasUser>
        <domain:hasObject>
          <domain:Content rdf:ID="contenidos">
            <hasEvents>
              <domain:Event rdf:ID="asignarcontenido"/>
            </hasEvents>
          </domain:Content>
        </domain:hasObject>
        <domain:securityCategory rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Browsing</domain:securityCategory>
     </domain:AccessCategory>
    </pattern:solution>
   <pattern:problem rdf:resource="#contenidos"/>
    <pattern:solution rdf:resource="#contenidos"/>
    <pattern:solution rdf:resource="#usuario"/>
  </hdp:HypermediaPattern>
```

A.1.6. [AS1]Access Control

Finalmente, los sistemas hipermedia deberían soportar políticas de seguridad que ofrezcan diferentes capacidades de manipulación a los usuarios según sus necesidades y responsabilidades en un determinado contexto. El siguiente conjunto de patrones, recogidos bajo la categoría `Access Control`, presenta diferentes niveles de seguridad.

[AS1]Access Control

Contexto Se está diseñando una [A]Hypermedia Application donde existen diferentes tipos de usuarios que pueden estar recogidos mediante [MZ1]Role Subtype. Tanto la información como los servicios definidos en la [AE1]User-centered Structure deberían tener un acceso controlado.

Problema ¿Cómo hacer para que el usuario pueda acceder de manera controlada al sistema y esté autorizado para realizar ciertas tareas?

Discusión 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 [Fernández *et al.*, 1996] 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.).

Relacionado con El control de acceso a los recursos es descrito en [MS1]Authorization. Una especialización de este patrón se encuentra en

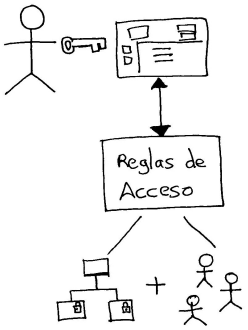


Figura A.32: Controlar el acceso a los usuarios para que accedan sólo a aquellos recursos para los cuales tengan permiso

`[MS2]Role Based Access Control`. Además se pueden asignar diferentes niveles de manipulación a la información con `[MS3]Multilevel Security`.

Fuentes [9]

Formalización

```
<hdp:HypermediaPattern rdf:ID="AS1">
   <pattern:context rdf:resource="AE1#AE1"/>
   <pattern:name>Access Control</pattern:name>
   <hdp:usesTo rdf:resource="MS2#MS2"/>
   <hdp:usesTo rdf:resource="MS3#MS3"/>
   <pattern:discussion>qué recursos pueden acceder y qué funciones se les permite
    realizar sobre los datos</pattern:discussion>
   <pattern:discussion> proporcionan servicios más avanzados a usuarios
    autorizados</pattern:discussion>
   <hdp:level>High</hdp:level>
   <pattern:context rdf:resource="MZ1#MZ1"/>
   <pattern:problem>acceder de manera controlada al sistema y esté autorizado
    para realizar ciertas tareas</pattern:problem>
   <pattern:context> tener un acceso controlado</pattern:context>
   <pattern:discussion>El control de acceso es un requerimiento esencial en los
    sistemas hipermedia </pattern:discussion>
   <pattern:discussion>mantener el control sobre la información
    personal</pattern:discussion>
   <hdp:usesTo rdf:resource=" MS1# MS1"/>
   <pattern:context rdf:resource="A#A"/>
   <hdp:aspect>Security</hdp:aspect>
   <pattern:discussion>definir las limitaciones de los usuarios en términos de lo
    que pueden hacer</pattern:discussion>
    <pattern:problem>
     <domain:Hyperdocument rdf:ID="sistema"/>
   </pattern:problem>
   <pattern:problem>
     <domain:User rdf:ID="usuario"/>
   </pattern:problem>
 </hdp:HypermediaPattern>
```

[MS1]Authorization

Contexto Se está definiendo el `[AS1]Access Control` para la aplicación y se desea controlar el acceso tanto a la información como a los servicios ofrecidos por la aplicación.

Problema ¿Cómo se puede describir quién está autorizado a acceder a los recursos del sistema?

Discusión Autorización es el acto de comprobar si un usuario tiene el permiso adecuado para acceder a un determinado dato o realizar una determinada acción, asumiendo que el usuario se ha autenticado de manera exitosa. La autorización depende de reglas específicas o listas de control de acceso presentadas por los administradores de la aplicación web o el propietario de los datos. Los permisos concedidos (autorizaciones) para acceder a los elementos protegidos necesitan ser indicados explícitamente, sino cualquier usuario podría acceder a cualquier recurso. La estructura de autorización debe ser independiente del tipo de recursos, p.e., debería describir accesos por usuarios, por programas, etc., de una manera uniforme.

Solución Representar reglas de autorización que definan el tipo de acceso de un usuario a un determinado recurso [Aedo *et al.*, 2003]:

Regla de acceso = (sujeto, objeto, categoría de acceso)

- Un sujeto es una entidad activa que puede ejecutar acciones sobre los objetos (por ejemplo recuperar un nodo, modificar un enlace o crear un nuevo contenido). Los sujetos son los usuarios de la aplicación, y teniendo en cuenta que otras entidades pueden activar acciones, como los programas o scripts, éstos heredan las mismas reglas que se aplican a los usuarios que las activan.
- Un objeto es una entidad pasiva que recibe los efectos de las acciones ejecutadas por los sujetos. Para mantener un razonable nivel de eficiencia, tanto en tiempo de ejecución como durante el proceso de administración, la granularidad de los objetos debe ser cuidadosamente determinada y, por esta razón solamente los nodos y los contenidos son considerados como objetos. Como los enlaces pueden ser definidos como conexiones entre un origen y un destino en un nodo o

contenido, ellos heredan las reglas que se apliquen a los elementos que componen su definición. Por ejemplo, un enlace es solamente presentado a un usuario cuando puede acceder tanto a su origen como a su destino.

- Una categoría expresa las posibilidades de acceso para una determinada aplicación. Cuatro categorías son consideradas:
  - No acceso: rechaza el acceso a un nodo o contenido
  - Visualización: recupera la información al seleccionar los enlaces o al usar otros mecanismos como índices o motores de búsqueda.
  - Personalización: añade a la categoría de Visualización la capacidad de incluir elementos personales, como contenidos, nodos o enlaces privados; y
  - Edición: añade a la Personalización la capacidad de modificar los elementos de la aplicación, es decir, manipular los elementos que son accedidos por todos los usuarios de la aplicación.

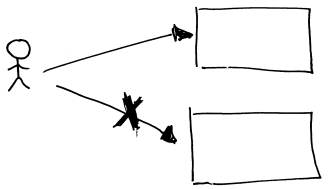


Figura A.33: Asignar a cada usuario una categoría de seguridad

Relacionado con El patrón `[MS2]Role Based Access Control` es una especialización de este patrón.

Fuentes [Fernández y Pan, 2001], [Aedo *et al.*, 2003]

Formalización

```
<hdp:HypermediaPattern rdf:about="# MS1">
    <pattern:discussion>La estructura de autorización debe ser independiente del
     tipo de recursos</pattern:discussion>
    <pattern:context rdf:resource=" AZ1# AZ1"/>
    <pattern:context>controlar el acceso tanto a la información como a los
     servicios ofrecidos por la aplicación</pattern:context>
    <hdp:level>Medium</hdp:level>
    <pattern:problem>quién está autorizado a acceder a los recursos del
     sistema</pattern:problem>
    <pattern:discussion>La autorización depende de reglas específicas o listas de
     control de acceso</pattern:discussion>
    <pattern:context rdf:resource="AS1#AS1"/>
    <hdp:specializedBy rdf:resource="MS2#MS2"/>
```

```
    <hdp:aspect>Security</hdp:aspect>
    <pattern:problem>
      <domain:User rdf:ID="usuario"/>
    </pattern:problem>
    <pattern:solution>
      <domain:AccessCategory rdf:ID="reglasautorizacion">
        <domain:hasUser rdf:resource="#usuario"/>
      </domain:AccessCategory>
    </pattern:solution>
  </hdp:HypermediaPattern>
```

[MS2]Role Based Access Control

Contexto Se ha definido diferentes reglas para cada usuario concreto con `[MS1]Authorization`, pero se pueden identificar conjuntos de usuarios que comparten similitudes.

Problema ¿Cómo asignar derechos a los usuarios según sus roles en la aplicación?

Discusión La mayoría de las instituciones tienen una variedad de categorías de trabajo que requieren de diferentes habilidades y responsabilidades. Por razones de seguridad los usuarios deberían conseguir sus derechos según su categoría de trabajo. Estas categorías profesionales pueden ser interpretadas como roles que las personas tienen para realizar sus obligaciones.

En concreto, en las aplicaciones web existen una variedad de usuarios, los cuales tienen diferentes necesidades para acceder a la información, según sus funciones. Pero asignar de manera individual los derechos de acceso requiere almacenar muchas reglas de autorización lo que dificulta las tareas de mantenimiento del administrador.

Solución Interpretar los usuarios como roles. El control de acceso basado en roles (RBAC *Role-Based Access Control*) es un mecanismo de seguridad que regula el acceso a los recursos basado en entidades organizativa basadas en roles. La política de control de acceso está embebida en los componentes de RBAC como relaciones usuario-rol, rol-permiso, y rol-rol. Estos componentes determinan si un determinado tipo de usuario se le permite acceder a una determinada información.

Patrones relacionados Los usuarios tienen que estar organizados en roles como se describe en el patrón `[MZ1]Role Subtype`.

Si se quieren añadir diferentes niveles de seguridad a los recursos véase el patrón `[MS3]Multilevel Security`.

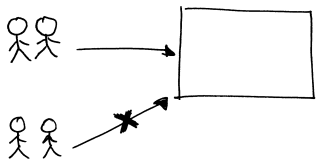


Figura A.34: Los usuarios de la aplicación recibirán el tipo de acceso del rol al que fueron asignados

Fuentes [Fernández y Pan, 2001]

Formalización

```
<hdp:HypermediaPattern rdf:ID="MS2">
   <pattern:context>identificar conjuntos de usuarios que comparten
    similitudes</pattern:context>
    <pattern:context rdf:resource="MS1#MS1"/>
   <hdp:aspect>Security</hdp:aspect>
   <pattern:discussion>variedad de categorías de trabajo que requieren de
    diferentes habilidades </pattern:discussion>
   <pattern:problem>asignar derechos a los usuarios según sus roles en la
    aplicación</pattern:problem>
   <hdp:usesTo rdf:resource="MZ1#MZ1"/>
   <pattern:discussion>asignar de manera individual los derechos de acceso
    requiere almacenar muchas reglas de autorización</pattern:discussion>
   <pattern:discussion>los usuarios deberían  conseguir sus derechos según su
    categoría de trabajo</pattern:discussion>
   <pattern:name>Role Based Access Control</pattern:name>
   <hdp:level>Medium</hdp:level>
   <pattern:problem>
     <domain:User rdf:ID="usuario">
      <domain:userList>
         <domain:User rdf:ID="concretos"/>
       </domain:userList>
       <domain:userType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >role</domain:userType>
     </domain:User>
   </pattern:problem>
   <pattern:solution>
     <domain:AccessCategory rdf:ID="permiso">
       <domain:hasUser rdf:resource="#usuario"/>
     </domain:AccessCategory>
   </pattern:solution>
 </hdp:HypermediaPattern>
```

[MS3]Multilevel Security

Contexto Se han definido las políticas de seguridad con `[MS1]Authorization` o `[MS2]Role-Based Access Control`, pero en algunos entornos los datos tienen diferentes niveles de sensibilidad, por ejemplo edición o visualización.

Problema ¿Cómo decidir el acceso a un entorno con clasificaciones de seguridad?

Discusión Es necesario proteger la confidencialidad y la integridad de los datos basada en su sensibilidad. Los usuarios tienen autorizaciones y pueden acceder a los documentos basadas en ellas.

Solución Asignar tanto a los usuarios como a la información (nodos y contenidos) clasificaciones o acreditaciones. Las clasificaciones incluyen niveles como navegación, personalización y edición. Para accesos confidenciales de los usuarios a los datos utilizar las definidas por el modelo Bell- LaPadula, mientras que para la integridad las reglas son definidas por el modelo de Biba.

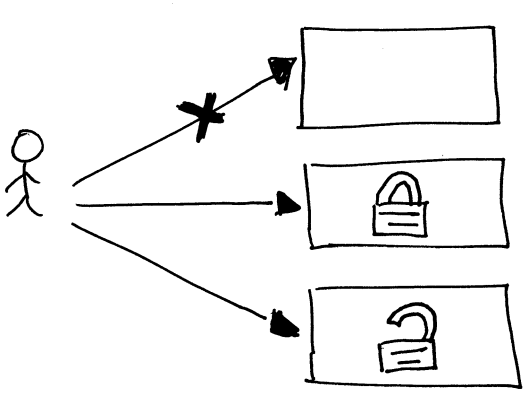


Figura A.35: Asignar acreditaciones a los usuarios y a la información

Relacionado con

Fuentes [Fernández y Pan, 2001]

Formalización

```
<hdp:HypermediaPattern rdf:ID="MS3">
   <pattern:discussion> proteger la confidencialidad y la integridad de los datos
    basada en su sensibilidad</pattern:discussion>
   <hdp:level>Medium</hdp:level>
   <pattern:context rdf:resource="MS2#MS2"/>
   <hdp:aspect>Security</hdp:aspect>
   <pattern:problem>decidir el acceso a un entorno con clasificaciones de
    seguridad</pattern:problem>
   <pattern:name>Multilevel Security</pattern:name>
   <pattern:context rdf:resource="MS1#MS1"/>
   <pattern:context>los datos tienen diferentes niveles de
    sensibilidad</pattern:context>
   <pattern:solution>
```

```
      <domain:AccessCategory rdf:ID="acceso">
        <domain:hasUser>
          <domain:User rdf:ID="usuario"/>
        </domain:hasUser>
      </domain:AccessCategory>
    </pattern:solution>
    <pattern:solution>
      <domain:Node rdf:ID="nodos">
        <domain:securityCategory rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Browsing</domain:securityCategory>
      </domain:Node>
    </pattern:solution>
    <pattern:problem rdf:resource="#acceso"/>
</hdp:HypermediaPattern>
```

Apéndice B

Ontología para patrones de diseño hipermedia

Una ontología de representación del conocimiento recoge las primitivas de modelado utilizadas para formalizar conocimiento (v.g. clases, relaciones, atributos, etc.). OWL [24] está destinado para la publicación y compartición de ontologías para la web y ha sido elegido como lenguaje de representación final para las ontologías descritas en este documento.

A continuación, se realiza una breve introducción a la lógica de descripción y su terminología (véase la sección B.1), la cual se ha utilizado como formalismo para describir las ontologías del capítulo 5, y a la correspondencia existente entre este formalismo y el lenguaje OWL (véase la sección B.2). Finalmente, se recogen las ontologías completas descritas en OWL para la descripción del concepto patrón de diseño hipermedia (véase la sección B.3).

B.1. Introducción a DL (*Description Logic*)

DL es un formalismo de representación del conocimiento mediante la definición de los conceptos importantes del dominio y sus propiedades a partir de los cuales se describen los individuos de dicho dominio. En DL, la base del conocimiento es un par

$$\sum = \langle T, A \rangle$$

donde T se denomina Tbox (*Terminological Box*) y A es Abox (*Assertional Box*). Adicionalmente, el tipo de lenguaje de descripción define los términos y los operadores para construir las expresiones.

El *TBox*, la parte terminológica, es un conjunto de axiomas que describen de manera intensional el conocimiento general sobre el dominio del problema, de la siguiente forma:

$$\begin{aligned} C &\sqsubseteq D(inclusión\,de\,conceptos) \\ C &\doteq D(definición\,de\,conceptos) \\ R &\sqsubseteq S(inclusión\,de\,roles) \\ R &\doteq S(definición\,de\,roles) \\ R^{+} &\sqsubseteq R(Transitividad\,de\,roles) \end{aligned}$$

donde C y D son conceptos, y R y S son las relaciones entre dichos conceptos.

El ABox es un conjunto de axiomas sobre aserciones en la cual los conceptos y roles describen de manera extensional el conocimiento específico de un problema particular de la siguiente forma:

$x \in D$ (instanciación del concepto)

$\langle x, y \rangle \in R$ (instanciación de relación)

B.2. Introducción a OWL (*Web Ontology Language*)

OWL es un lenguaje de ontología desarrollado por el grupo de trabajo *W3C Web Ontology* (WebOnt) y en este momento es recomendación del consorcio W3 [24]. OWL se define como una extensión a RDF en la forma de un vocabulario entailment, es decir la sintaxis de OWL es la sintaxis de RDF(S) y su semántica es una extensión de la semántica de RDF. Se han definido tres subconjuntos con diferentes expresividad:

OWL *Lite* extiende RDF(S) y recoge las características comunes de OWL y está pensado para crear taxonomías de clases y restricciones simples.

Constructor	Sintaxis DL
complementOf	$\neg C$
intersectionOf	$C \sqcap D$
unionOf	$C \sqcup D$
oneOf	$\{x_1, x_2, ..., x_n\}$
allValuesFrom	$\forall R.C$
someValuesFrom	$\exists R.C$
hasValue	$\exists \{R.x\}$
cardinality	$= nR$
maxCardinality	$\leq nR$
minCardinality	$\geq nR$

Tabla B.1: Constructores de clase OWL y su sintaxis en DL

Axioma	Sintaxis DL
subClassOf	$C \sqsubseteq D$
equivalentClass	$C \equiv D$
disjointWith	$C \sqsubseteq \neg D$
enumeratedClass	$A \equiv \{x_1, x_2, ..., x_n\}$
subPropertyOf	$R \sqsubseteq S$
domain	$\geq 1R \sqsubseteq C$
range	$\top \sqsubseteq \forall R.C$
inverseOf	$R \equiv S_{-}$
SummetricProperty	$R \equiv R_{-}$
TransitiveProperty	$R^{+} \sqsubseteq R$
FunctionalProperty	$\top \sqsubseteq \leq 1R$
inverseFunctionalProperty	$\top \sqsubseteq \leq 1R^{-}$
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$
differentFrom	$\{x_1\} \equiv \neg \{x_2\}$

Tabla B.2: Axiomas OWL y su sintaxis en DL

OWL DL incluye el vocabulario completo de OWL es un lenguaje SHOIN(D) según la terminología de la lógica de descripción [Horrocks y Patel-Schneider, 2003].

OWL *Full* brinda el máximo poder expresivo posible, a costa de no poder siempre dar una respuesta ante consultas hechas al modelo.

Hay una correspondencia entre OWL y la base de conocimiento de DL [Horrocks y Patel-Schneider, 2003]. En la tabla B.1 se recoge la correspondencia entre la sintaxis abstracta de OWL para la creación de clases y su sintaxis asociada en DL, y en la tabla B.2 la sintaxis abstracta de OWL para la definición de axiomas y su correspondiente sintaxis en DL.

B.3. Hypermedia Design Pattern

Esta ontología se basa en la importación modular de las ontologías componentes de un patrón (véase la sección B.3.1 y dominio hipermedia (véase la sección B.3.2), para describir un patrón de diseño hipermedia. Además, se puede encontrar publicada en http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:hypermedia="http://hypatterns.no-ip.info/ontologies/hypermedia.owl#"
    xmlns:pattern="http://hypatterns.no-ip.info/ontologies/pattern.owl#"
    xmlns="http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl#"
  xml:base="http://hypatterns.no-ip.info/ontologies/hypermediaPattern.owl"'>
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl"/>
    <owl:imports rdf:resource="http://hypatterns.no-ip.info/ontologies/hypermedia.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="HypermediaPattern">
    <rdfs:label>Hypermedia Design Pattern</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl#Pattern"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="usesTo">
    <rdfs:subPropertyOf rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl#relatedTo"/>
    <rdfs:range rdf:resource="#HypermediaPattern"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="suggestsTo">
    <rdfs:subPropertyOf rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl#relatedTo"/>
    <rdfs:range rdf:resource="#HypermediaPattern"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="specializedBy">
    <rdfs:subPropertyOf rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl#relatedTo"/>
    <rdfs:range rdf:resource="#HypermediaPattern"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="level">
    <rdfs:domain rdf:resource="#HypermediaPattern"/>
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Medium</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Low</rdf:first>
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            </rdf:rest>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >High</rdf:first>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="annotation">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AnnotationProperty"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="aspect">
    <rdfs:domain rdf:resource="#HypermediaPattern"/>
```

```
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:parseType="Resource">
                <rdf:rest rdf:parseType="Resource">
                  <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >Security</rdf:first>
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >Interaction</rdf:first>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                  </rdf:rest>
                </rdf:rest>
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Personalization</rdf:first>
              </rdf:rest>
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Presentation</rdf:first>
            </rdf:rest>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Structure</rdf:first>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Navigation</rdf:first>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
  </owl:DatatypeProperty>
<rdf:Description rdf:about="http://hypatterns.no-ip.info/ontologies/pattern.owl#Actor">
    <owl:equivalentClass>
      <rdf:Description rdf:about="http://hypatterns.no-ip.info/ontologies/hypermedia.owl#Component">
        <owl:equivalentClass rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl#Actor"/>
      </rdf:Description>
    </owl:equivalentClass>
  </rdf:Description>
<rdf:Description rdf:about="http://hypatterns.no-ip.info/ontologies/pattern.owl#Actor">
    <owl:equivalentClass>
      <rdf:Description rdf:about="http://hypatterns.no-ip.info/ontologies/hypermedia.owl#Hyperdocument">
        <owl:equivalentClass rdf:resource="http://hypatterns.no-ip.info/ontologies/pattern.owl#Actor"/>
      </rdf:Description>
    </owl:equivalentClass>
  </rdf:Description>
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 1.3, Build 225.4)  http://protege.stanford.edu -->
```

B.3.1. Pattern

Esta ontología recoge los conceptos y relaciones que representan el formato de un patrón de diseño. Además, se puede encontrar publicada en http://hypatterns.no-ip.info/ontologies/pattern.owl.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://hypatterns.no-ip.info/ontologies/pattern.owl#"
  xml:base="http://hypatterns.no-ip.info/ontologies/pattern.owl">
  <owl:Ontology rdf:about=""/>
```

```
<owl:Class rdf:ID="Actor"/>
<owl:Class rdf:ID="Pattern">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:ID="context"/>
      </owl:onProperty>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Pattern"/>
            <owl:Restriction>
              <owl:someValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
              <owl:onProperty>
                <owl:DatatypeProperty rdf:ID="intention"/>
              </owl:onProperty>
            </owl:Restriction>
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Actor"/>
            <owl:Restriction>
              <owl:onProperty>
                <owl:DatatypeProperty rdf:ID="motivation"/>
              </owl:onProperty>
              <owl:someValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
            </owl:Restriction>
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="problem"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="solution">
  <rdfs:range rdf:resource="#Actor"/>
  <rdfs:domain rdf:resource="#Pattern"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#problem">
  <rdfs:domain rdf:resource="#Pattern"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="discussion">
  <rdfs:domain rdf:resource="#Pattern"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="image">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
  <rdfs:domain rdf:resource="#Pattern"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#motivation">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="diagram">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
  <rdfs:domain rdf:resource="#Pattern"/>
</owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:about="#intention">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:about="#context">
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:ID="relatedTo"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Pattern"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#relatedTo">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Pattern"/>
    <owl:inverseOf rdf:resource="#context"/>
  </owl:TransitiveProperty>
  <owl:InverseFunctionalProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#Pattern"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:InverseFunctionalProperty>
</rdf:RDF>
```

B.3.2. Hypermedia

Esta ontología recoge los conceptos y relaciones que representa las entidades de diseño a partir de las cuales se puede definir la estructura estática de una aplicación hipermedia. Además, se puede encontrar publicada en http://hypatterns.no-ip.info/ontologies/hypermedia.owl.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://hypatterns.no-ip.info/ontologies/hypermedia.owl#"
  xml:base="http://hypatterns.no-ip.info/ontologies/hypermedia.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Anchor">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class>
           <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:ID="Node"/>
              <owl:Class rdf:ID="Content"/>
            </owl:unionOf>
          </owl:Class>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="location"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Alignment">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Constraint"/>
```

```
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AccessCategory">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
       <owl:ObjectProperty rdf:ID="hasObject"/>
      </owl:onProperty>
      <owl:someValuesFrom>
        <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Node"/>
            <owl:Class rdf:about="#Content"/>
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Event"/>
<owl:Class rdf:ID="Synchronization">
  <rdfs:subClassOf rdf:resource="#Constraint"/>
</owl:Class>
<owl:Class rdf:ID="Attribute"/>
<owl:Class rdf:ID="Root">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="CompositeComponent"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="User">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Component"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Node">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Hyperdocument">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#AccessCategory"/>
        <owl:Class rdf:about="#Anchor"/>
        <owl:Class rdf:about="#Attribute"/>
       <owl:Class rdf:about="#Component"/>
        <owl:Class rdf:about="#Constraint"/>
        <owl:Class rdf:about="#Event"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component">
  <rdfs:label>Hypermedia Elements</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Content">
  <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
<owl:Class rdf:ID="ContextualLink">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Link"/>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:about="#Link">
  <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
<owl:Class rdf:about="#CompositeComponent">
  <rdfs:subClassOf rdf:resource="#Component"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Node"/>
            <owl:Class rdf:about="#Content"/>
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasComponents"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasTarget">
  <rdfs:range rdf:resource="#Anchor"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Link"/>
        <owl:Class rdf:about="#Constraint"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSourceConstraint">
  <rdfs:range rdf:resource="#Content"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasUser">
  <rdfs:range rdf:resource="#User"/>
  <rdfs:domain rdf:resource="#AccessCategory"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasComponents">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Node"/>
        <owl:Class rdf:about="#CompositeComponent"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <rdfs:domain rdf:resource="#CompositeComponent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLocation">
  <rdfs:range rdf:resource="#Content"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAnchors">
  <rdfs:range rdf:resource="#Anchor"/>
  <rdfs:domain rdf:resource="#Node"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConstraints">
  <rdfs:domain rdf:resource="#Node"/>
  <rdfs:range rdf:resource="#Constraint"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAttributes">
  <rdfs:domain rdf:resource="#Component"/>
  <rdfs:range rdf:resource="#Attribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasObject">
```

```
  <rdfs:domain rdf:resource="#AccessCategory"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="context">
  <rdfs:range rdf:resource="#Component"/>
  <rdfs:domain rdf:resource="#ContextualLink"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEvents">
  <rdfs:domain rdf:resource="#Component"/>
  <rdfs:range rdf:resource="#Event"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAnchor"/>
<owl:ObjectProperty rdf:ID="hasElements">
  <rdfs:range rdf:resource="#Component"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#location">
  <rdfs:domain rdf:resource="#Anchor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSource">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Link"/>
        <owl:Class rdf:about="#Constraint"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#Anchor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasContents">
  <rdfs:domain rdf:resource="#Node"/>
  <rdfs:range rdf:resource="#Content"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="userType">
  <rdfs:domain rdf:resource="#User"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >group</rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </rdf:rest>
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >individual</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="#Attribute"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="action">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Event"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="begin">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Synchronization"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="type">
  <rdfs:domain rdf:resource="#Synchronization"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="value">
  <rdfs:domain rdf:resource="#Attribute"/>
```

```
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="mediatype">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="relationType">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Generalization</rdf:first>
        </rdf:rest>
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Aggregation</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="end">
  <rdfs:domain rdf:resource="#Synchronization"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPosition"/>
<owl:DatatypeProperty rdf:ID="direction">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Bydirect</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >None</rdf:first>
            </rdf:rest>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >To</rdf:first>
        </rdf:rest>
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >From</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="anchorID">
  <rdfs:domain rdf:resource="#Anchor"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="securityCategory">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Editing</rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Personalizing</rdf:first>
        </rdf:rest>
```

```
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Browsing</rdf:first>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Node"/>
          <owl:Class rdf:about="#Content"/>
          <owl:Class rdf:about="#CompositeComponent"/>
          <owl:Class rdf:about="#AccessCategory"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="relationship">
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >generalization</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >aggregation</rdf:first>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
    <rdfs:domain rdf:resource="#CompositeComponent"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="condition">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Event"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="UID">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Component"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:ID="userList">
    <rdfs:domain rdf:resource="#User"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#User"/>
  </owl:TransitiveProperty>
  <owl:InverseFunctionalProperty rdf:ID="hasPresentationSpecification">
    <rdfs:domain rdf:resource="#Component"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Attribute"/>
  </owl:InverseFunctionalProperty>
</rdf:RDF>
```

Apéndice C

Cuestionarios de evaluación

C.1. Cuestionario para la evaluación del lenguaje de pdh

Evaluación de un lenguaje de patrones de diseño hipermedia

Te agradeceríamos que rellenases este formulario cuyo objetivo es analizar la usabilidad y utilidad del lenguaje de patrones en la generación de un modelado conceptual.

DATOS DEL EVALUADOR

Titulación que estudias:_____
Edad: _____

Conocimientos en diseño:

	Mucho	Algo	Neutro	Poco	Nada
Familiaridad con el diseño de aplicaciones: Software Hipermedia/Web					
Familiaridad con el concepto de patrón de diseño					
Familiaridad con la utilización de patrones Catálogo/Lenguaje _____ Catálogo/Lenguaje _____ Catálogo/Lenguaje _____ Catálogo/Lenguaje _____	....	....	....	....	....
Tiempo que has dedicado al lenguaje de patrones de diseño hipermedia					

Si deseas volver a participar en próximas evaluaciones escribe tu e- mail:
_____

Formulario de Evaluación

¿Qué opinión te merece el lenguaje con respecto a las siguientes características?

	Muy buena	Buena	Neutra	Mala	Muy mala
Utilidad					
Completitud					
Organización					
Estilo narrativo					
Formato					
Representación de la solución					

¿Qué opinión te merece el lenguaje con respecto a las siguientes características?

		Muy buena	Buena	Neutra	Mala	Muy mala
[AE1]User-centered Structure	Utilidad					
	Completitud					
[AN1]Multiple Ways to Navegate	Utilidad					
	Completitud					
[AP1]Aesthetics	Utilidad					
	Completitud					
[AI1]Interaction	Utilidad					
	Completitud					
[AZ1]Personalization	Utilidad					
	Completitud					
[AS1]Access Control	Utilidad					
	Completitud					

¿Has encontrado algún aspecto que mejorarías? (Eliminar algún patrón, añadir, reescribir, nombre del patrón etc.)

Aspecto	Mejora

¿Has tenido problemas a la hora de identifcar el patrón que cubría a cada requisito? ¿Cuáles?

¿Has tenido problemas a la hora de transformar la solución del patrón a una estructura de diseño concreta? ¿Cuáles?

¿Estás satisfecho con el resultado final del modelado?

Completamente	Bastante	Medio	No mucho	Nada

¿Hubiéses preferido realizar el modelado de la aplicación sin el uso del lenguaje de patrones?

Completamente	Bastante	Medio	No mucho	Nada

¿Has utilizado el nombre de los patrones para comunicarte con el resto de los miembros del equipo, por ejemplo para discutir cuestiones sobre el diseño?

Completamente	Bastante	Medio	No mucho	Nada

Comentarios sobre los patrones y su uso

C.2. Cuestionario para la evaluación de AnnotPat

MN1. Index Navigation

Nombre: Index Navigation
Contexto: La información ha sido organizada mediante [AE1] Hierarchical Organization y es necesario poder acceder a ella.
Problema: ¿Cómo proporcionar acceso rápido a un grupo de nodos para que los usuarios que están interesados en uno o más de ellos sean capaces de seleccionar alguno?
Discusión: Este requisito es menos obvio de lo que pueda parecer a primera vista; de hecho, en muchas situaciones, el usuario no es capaz de hacer una elección ya que pueden no conocer suficientemente bien el tema de la aplicación, no ser capaz de identificar los elementos en la lista, o no tener una meta específica en mente. Además uno de los inconvenientes de la navegación puede ser la necesidad de volver a un punto de partida para acceder a otro nodo, necesitando así dos pasos para moverse de un nodo a otro.
Por lo tanto, es necesario hacer accesible la información al usuario para que pueda decidir.
Solución: Define enlaces desde el punto de entrada al conjunto de nodos a cada uno de sus miembros, y de cada miembro al punto de entrada. Para acelerar la navegación, mantén siempre accesible el punto de entrada que contiene los enlaces a cada miembro para que el usuario pueda acceder directamente a cualquiera de ellos sin tener que volver al punto de entrada.
El punto de entrada o índice es una lista de anclas, enlaces o nodos ordenados por el alfabeto, el tema, el autor, la materia, etc. Esta definición incluye herramientas comunes en papel como índices alfabéticos, tabla de contenidos, glosarios y otros.

Diagrama: indexnavigationpattern.png

Relacionado con: BN1.Collection Center define qué contenidos debería contener el punto de entrada o índice.
Si quieres ayudar a los usuarios a moverse por los elementos de la colección, siguiendo algún orden utiliza MN2.Guided Tour.
Para aquellos usuarios que pueden no conocer bien los contenidos de la aplicación o no son capaces de identificar los elementos de una lista o no tienen todavía una meta específica en mente necesitan de otras herramientas de ayuda a la navegación como BN2.SiteMap o BN3.Search Action Module.
Además si este patrón ha sido utilizado de manera recursiva para acceder al espacio de información de manera jerárquica, utiliza BN1.Location Bread Crumbs para orientar al usuario sobre dónde se encuentra.
Para adaptar este mecanismo de navegación a diferentes contextos, utiliza MN3.Navigational Context.

Descripción detallada de las tareas a realizar

En concreto, durante tu trabajo nos gustaría que llevaras a cabo las siguientes tareas en Annotpat:

1. Creación de un nuevo repositorio
   1.1. Especificar un identificador
2. Creación de un nuevo patrón (*Index Navigation*)
   2.1. Especificar un identificador
   2.2. Elegir el orden de los campos del patrón
   2.3. Rellenar cada uno de los campos con el texto correspondiente
   2.4. Copiar/Pegar texto
3. Creación de anotaciones
   3.1. Seleccionar texto
   3.2. Realizar una anotación
4. Borrado de anotaciones
   4.1. Hacer clic sobre la anotación
   4.2. Borrar la anotación
5. Guardar el repositorio
6. Cerrar el repositorio

Caso de uso:

AN1.Multiple Ways to Navigate

Contexto: Se está construyendo una A.Hypermedia Application y la información ha sido organizada siguiendo una AE1.User-centered structure. 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?:

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. Quizás 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.

Y a veces, los usuarios no quieren perder el tiempo aprendiendo a navegar por el sistema, sólo quieren entrar, conseguir la información y salir tan rápido como sea posible.

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 grandes 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.

Diagrama: hypermediaaplicationpattern.jpg

Relacionado con: El patrón MN1.Index Navigation ayuda a los usuarios a seleccionar un destino, mientras que el patrón MN2.Guided Tour Navigation 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.Navigational Context. 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.Location Bread Crumbs. Para mostrar la situación del usuario con respecto al espacio de información total de la aplicación, utilizar el patrón BN2. Site Map. Para que el usuario pueda conseguir la información deseada de forma directa, se puede proporcionar una BN3.Search Action Module. Los usuarios deberían ser capaces de volver a la página principal o a otros puntos mayores de navegación en un sólo paso con BN4.One Jump Home.

1. Evaluación de AnnotPat

Te agradeceríamos que rellenases este formulario cuyo objetivo es analizar la usabilidad y utilidad de AriadneTool en el diseño de aplicaciones hipermedia.

FORMULARIO DE EVALUACIÓN

I. DATOS DEL EVALUADOR

Conocimientos en patrones de diseño:

	Mucho	Algo	Neutro	Poco	Nada
Familiaridad con patrones de diseño					
Familiaridad con patrones de diseño hipermedia					

Conocimientos en ontologías:

	Mucho	Algo	Neutro	Poco	Nada
Familiaridad con ontologías					
Familiaridad con editores de ontologías					
Herramienta	.....	.....	.....	.....	.....
_____	.....	.....	.....	.....	.....
Herramienta	.....	.....	.....	.....	.....
_____					
Herramienta					
_____	.....	.....	.....	.....	.....
	.....	.....	.....	.....	.....
Familiaridad con herramientas de anotaciones semánticas	.....	.....	.....	.....	.....
Herramienta					
_____					
Herramienta					
_____					
Herramienta					
_____					

Si deseas volver a participar en próximas evaluaciones de la herramienta escribe tu e- mail:_____

Vuelve a crear un nuevo repositorio y un nuevo patrón para *Multiple Ways to Navigate*

7. Recuperación de un patrón
   7.1. Abrir el patrón *Index Navigation*
8. Visualización de un patrón
   8.1. Seleccionar de la lista de patrones *Index Navigation*
9. Ocultación de un patrón
   9.1. Deseleccionar de la lista de patrones *Index Navigation*

Enlaza el patrón *Multiple Ways to Navigate* con *Index Navigation*

10. Exportar el repositorio al servidor
    10.1. Visualizar los patrones conectándose a:
    http://hypatterns.no-ip.info/

11. Recuperar un repositorio
12. Cerrar la aplicación

Para comparar la utilidad y usabilidad de Annotpat con respecto a otros modos de crear instancias de patrones a partir de las ontologías definidas para su formalización, te proponemos la siguiente tarea:

13. Crear un patrón con un editor de ontologías Protégé
    13.1. Abrir el proyecto *hypermediaPattern.pprj*
    13.2. Seleccionar la pestaña *Individuals*
    13.3. Crear un individuo del tipo *HypermediaPattern* y llamarlo *IndexNavigation*
        13.3.1. Para cada uno de los campos que forman parte del individuo crear instancias cuyo valor sean las anotaciones realizadas con AnnotPat

I. **EVALUACIÓN DE LA INTERFAZ DE USUARIO DE ANNOTPAT**

Indica tu opinión sobre las siguientes características de la herramienta

	Muy buena	Buena	Neutra	Mala	Muy mala
Utilidad					
Rapidez					
Fiabilidad					
Facilidad de uso					
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					
Diseño del estado de las anotaciones					

Indica tu grado de satisfacción sobre el acceso a los diferentes elementos del programa a través del:

	Muy buena	Buena	Neutra	Mala	Muy mala
Menú					
Barra de botones					

¿Quieres hacer algún comentario adicional sobre la utilidad de AnnotPat?

¿Quieres hacer algún comentario más sobre el interfaz?

Indica tu grado de satisfacción global con respecto a la herramienta:

Muy safisfecho	Safisfecho	Indiferente	Decepcionado	Muy decepcionado

II. **EVALUACIÓN DETALLADA DE LAS TAREAS**

Tarea 1. Creación de un repositorio

Tras haber creado un nuevo repositorio con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 2. Creación de un patrón

Tras haber creado un nuevo patrón con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 3. Creación de anotaciones

Tras haber creado anotaciones con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 4. Borrado de anotaciones

Tras haber borrado anotaciones con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 5. Guardar un repositorio

Tras haber guardado un repositorio con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 7. Recuperación de un patrón

Tras haber recuperado un patrón con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 6. Cerrar un repositorio

Tras haber cerrado un repositorio con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 8. Visualización de un patrón

Tras haber visualizado un patrón con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 9. Ocultación de un patrón

Tras haber ocultado un patrón con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 10. Exportar un repositorio a un servidor

Tras haber exportado el repositorio para su visualización en un servidor con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 11. Recuperar un repositorio previamente guardado

Tras haber recuperado el repositorio realizado con AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 12. Cerrar la aplicación

Tras haber cerrado AnnotPat indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?

Tarea 13. Crear un patrón con el editor de ontologías Protégé

Tras haber creado un patrón con Protégé indica cómo te ha parecido esta tarea:

	Mucho	Algo	Neutro	Poco	Nada
Fácil de realizar					
Útil					
Consistente					
Tangible					
Fiable					

Expresa tu opinión sobre:

	Muy buena	Buena	Neutro	Mala	Muy mala
Consistencia de la interfaz					
Adecuidad de la interfaz al tipo de usuario					
Calidad de los iconos					
Adecuidad de los iconos					
Legibilidad					
Diseño de los menús					
Diseño de las barras de botones					

¿Has echado en falta alguna opción o acción complementaria? Si es así, indica cual(es) y por qué crees que serían de utilidad

Acción/Tarea	Posible utilidad

¿Has tenido problemas al realizar alguna acción?

Acción/Tarea	Problema

¿Quieres hacer algún comentario más sobre esta tarea?