Universidad Nacional de La Plata

Facultad de Informática

Trabajo de Grado

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Una Herramienta para el Diseño de Hipermedia.

Reusabilidad de Diseño en Aplicaciones Web.

Trabajo de grado para la Licenciatura en Informática de

Fernando Daniel Lyardet

Dirigida por: Dr. Gustavo H. Rossi

Abril 2002

TES
02/4
DIF-02201
SALA

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMATICA
Biblioteca
50 y 120 La Plata
catalogo.info.unlp.edu.ar
biblioteca@info.unlp.edu.ar

DIF-02201

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Indice

Índice de Ilustraciones

Dedicación

A mis Padres.


BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Agradecimientos

Agradezco al Dr.Gustavo Rossi, Director y amigo, que me animó a comenzar mi carrera de investigador.

Quiero expresar también mi gratitud para todos aquellos que durante estos años me han brindado su ayuda y amistad.

Glosario

ACID	Atomicity, Consistency, Isolation, and Durability
Affordance	Término técnico que refiere a las características de objetos - qué clase de operaciones y de manipulaciones se puede hacer a un objeto determinado
API	Application Program Interface
BMP	Bean Managed Persistence
BMT	Bean Managed Transaction
CMP	Container Managed Persistence
CMT	Container Managed Transaction
Contraste	Diferencia en luminiscencia entre un objeto y su fondo en una pantalla de visualización [Preece]
CORBA	Common Object Request Broker Architecture
DD	Deployment Descriptor
DTD	Document Type Definition
EAR	Enterprise Archive
EJB	Enterprise JavaBeans
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Security Socket Layer
IDE	Integrated Development Environment
IIOP	Internet Inter-ORB Protocol
J2EE	Java 2 Enterprise Edition
JAR	Java Archive
JDBC	Java Database Connectivity
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
JTA	Java Transaction API
LRU	Last Recently Used
OOHDM	Object-Oriented Hypermedia Design Method
RMI	Remote Method Invocation
SGBD	Sistema Gerenciador de Banco de Dados
SGBDOO	Sistema de Gerenciamento de Banco de Dados Orientado a Objetos

SQL	Structured Query Language
SSL	Security Socket Layer
UML	Unified Modeling Language
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAR	Web Archive
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations


BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

1 Introducción

Este capítulo conforma el marco de desarrollo del presente trabajo de grado. Las tres primeras secciones plantean el objetivo que se ha perseguido, su motivación y principales contribuciones, las que luego serán fundamentadas en los siguientes capítulos.

La cuarta sección presenta el estado del arte con respecto a los conceptos básicos sobre los que se ha trabajado, y al mismo tiempo explica qué faceta de cada uno es la que ha incidido en esta tesis. Cabe destacar que se suponen conocimientos básicos sobre el paradigma de orientación a objetos.

1.1 Objetivo del Trabajo

Estudiar la problemática del desarrollo de aplicaciones hipermediales desde el punto de vista del diseño, reusabilidad y documentación.

1.2 Motivación

Este trabajo surge como resultado de un proyecto para el estudio del diseño de aplicaciones hipermediales, la posibilidad de reusabilidad de modelos, software y experiencia a través de la utilización de patrones de diseño.

1.3 Contribuciones

Las contribuciones más importantes de este trabajo son las siguientes:

Una herramienta para diseñar con la notación OOHDM.

Un framework para aplicaciones web que implementa los conceptos de OOHDM.

Un catálogo de patterns de diseño de software.

Introduce la integración del modelado de los procesos.

Introduce una implementación para el soporte de procesos en aplicaciones web.

Extiende el Framework JHotDraw para soportar versiones del JDK 1.2 en adelante. Agrega soporte para Java2D.

Un framework para crear wizzards bajo Java Swing que generalmente son muy laboriosos y complejos de implementar.

20 publicaciones internacionales en Congresos y revistas con referato.

1.4 Hipertexto e Hipermedia

Tener acceso a un sistema remoto al otro lado del mundo en pocos segundos se ha convertido hoy en una trivialidad con la Web. Gracias a esta, se ha puesto a disposición una abundancia de información audio-visual y textual en línea como nunca antes. Las compras completamente automatizadas, acceso sobre cualquier información disponible a nivel mundial, realizar búsquedas sobre millones de documentos informativos se ha convertido en parte de nuestro ambiente cotidiano.

Hace aproximadamente diez años existía una duda genuina acerca de si alguna vez se lograría un estándar global del hipertexto sería ficción o algo realizable [Nielsen93]. Hoy, los documentos de HTML [w3] se escriben o generaran cada día en cantidades masivas.

El concepto de hipertexto e hipermedia no es nuevo. La primera noción del hipertexto data de 1945. Vannevar Bush previó una nueva profesión de armadores de rastros ("Trail Blazers"), que "encontrarían placer en la tarea de establecer rastros útiles a través de la masa enorme de registros comunes" [Bush45]. Esto tiene obviamente poco sentido en términos prácticos sin el uso de tecnología para llevar a cabo un procedimiento tan intensivo de búsqueda. Cuando esa tecnología llegó a estar disponible en los años 60, pronto una comunidad de investigación en hipertexto y hipermedia emergió. El expediente común se ha convertido en el WWW hoy, y el profesión de la "armador de rastros" es realizado por millones de usuarios.

Es bastante fácil distinguir hipertexto e hipermedia del texto ordinario [Nielsen93]. Los textos tradicionales, si están impresos o en formato de datos legible por computador, son secuenciales, significando que hay una sola secuencia lineal que define el orden en el cual el texto debe ser leído. Un lector está forzado a seguir esta secuencia, quiera o no. Lo mismo se aplica a las interfaces de usuario típicas de muchas aplicaciones específicas: el usuario tiene que utilizar la aplicación en un orden predefinido. En cambio, el hipertexto es no-secuencial. No se impone ningún orden de lectura al lector; el autor de una estructura hipertextual ofrece muchas alternativas cómo leerla. Tal estructura o red de hipertexto consiste en varios documentos, y cada una de las referencias que contienen el uno al otro. Un lector individual determinará a la hora de leer cuales de estas referencias seguirá. Si bien ésta es una definición de qué consiste un hipertexto, no refleja claramente su idea subyacente.

Una noción básica muy importante del hipertexto es que, al contrario de otras formas de presentar la información en informática, el usuario tiene recibir la mayor cantidad de información útil, poniendo a disposición del usuario la posibilidad de decidir si explorará o no esa información. Por consiguiente, el usuario debe también sentirse libre explorar cualquier información en cualquier sentido.

2 La Metodología de Diseño OOHDM.

2.1 Introducción

El campo de la hipermedia se encuentra en un rápido crecimiento. Nuevas áreas de aplicaciones, que necesitan de la flexibilidad de la hipermedia unido a una rica variedad de tipos de datos multimedia, está emergiendo. Estas aplicaciones abarcan desde sofisticados sistemas de información de acceso global (extranets) soportados sobre bases de datos orientadas a objetos o relacionales, pasando por ambientes de soporte de decisiones y de ingeniería de software, hasta sistemas de informaciones geográficas.

Desgraciadamente, la construcción de grandes aplicaciones de hipermedia es difícil debido principalmente, a la combinación de navegación controlada por el usuario y la naturaleza propia de la información multimedia.

Con la revolución de las comunicaciones y el advenimiento de Internet y el WWW, se impone un nuevo reto a los sistemas de información: son accedidos por los usuarios al rededor del planeta y usualmente estas aplicaciones no han sido diseñadas desde un punto de vista global. En este contexto, una sistema de información debería tener en cuenta nuevos y diversos aspectos:

Usuarios con diferentes necesidades de información: estudiantes, investigadores, inversionistas, maestros, etc.

Las aplicaciones debe responder a un nuevo modelo de interacción, donde las aplicaciones son accedidas en la misma forma que cualquier otro: navegando a través de links hipermediales.

Las aplicaciones son ahora accedidas globalmente, por lo tanto considerar diferentes modelos de usuarios (donde intervienen aspectos mucho mas sutiles como los étnicos, culturales y edad juegan un importante rol), se convierte en un problema central de diseño para un sistema de información

A continuación describimos a OOHDM, como un método de diseño para sistemas de información y aplicaciones hipermediales. Describiremos brevemente las características de esta metodología, y de que manera soluciona e integra en el proceso de diseño los nuevos y sofisticados requerimientos de un sistema de información global.

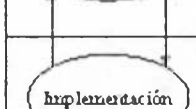
Etapas	Productos	Mecanismos	Objetivos
	Clases, subsistemas, relaciones, atributos perspectivas	Clasificación, composición, generalización y especialización	Modelizar la semántica del dominio de la aplicación
	Nodos, links, estructuras de acceso , contextos navegacionales, trasformaciones navegacionales.	El Mapeo entre los objetos conceptuales y navegacionales. Patrones de navegación para describir la estructura de toda la aplicación	Tiene en cuenta las características del usuario y tarea. Se enfatiza en aspectos cognitivos y arquitecturales.
	Objetos de la Interfaz Abstracta, respuesta a eventos externos, transformaciones de interfaz.	Mapeo entre los objetos de navegación y objetos perceptibles,	Modelizar los objetos perceptibles, implementar las metáforas elegidas. Describir la interfaz para los objetos navegacionales.
			
	Aplicación funcionando	Los provistos por el ambiente en que corre la aplicación (Toolbook, www,etc)	Performance, completitud

Figura 1. Descripción de la metodología y sus etapas.

2.2 Modelado Conceptual

El modelado conceptual es la actividad responsable del análisis del dominio de la aplicación, es decir, engloba todo el universo de informaciones relevantes para la aplicación en cuestión, aunque apenas un subconjunto de dichas informaciones fuera considera posteriormente en la implementación.

El resultado del modelado conceptual de una aplicación consiste de un esquema conceptual conteniendo los objetos del dominio de la aplicación (Clases, relaciones y subsistemas.

El Diagrama conceptual es un diagrama que contiene las clases representativas de la aplicación las relaciones existentes entre dichas clases y los subsistemas que agrupan las clases. Las clases serán utilizadas luego durante el Diseño Navegacional para derivar clases de Nodos, y las relaciones serán utilizadas para derivar Links.

Clases

Las clases en OOHDM poseen el mismo significado que las clases de una modelización de objetos y presentan las mismas características (atributos, relaciones y métodos).

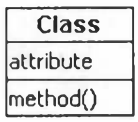


Figura 2. Notación de una Clase Conceptual.

Atributos

Los atributos de las clases representan las propiedades de los objetos (nombre, descripción, etc.).

Cada atributo debe ser de un tipo predefinido simple (e.g. String), complejo (e.g. Collection) o un tipo definido por la aplicación.

En OOHDM, los atributos pueden tener múltiples perspectivas, donde cada una de ellas significa una representación diferente para el contenido del atributo. Cuando un atributo tiene múltiples perspectivas, ellas son enumeradas entre "[" y "]", y cada una es representada por un identificador, seguido de ":" y un tipo.

Existen situaciones en las cuales las múltiples perspectivas no son utilizadas. La primera es cuando dicho atributo es presentado con la misma perspectiva cada vez que objeto sea mostrado. En esta situación, es necesario establecer la perspectiva *default*, que se la designa con "+". LA perspectiva *default* es la perspectiva en la cual el atributo tendrá un valor para todas las instancias, y recibe el nombre del atributo, mientras que el resto deben siempre precisar un identificador y pueden o no aparecer en las instancias.

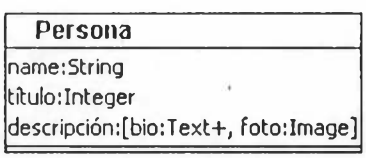


Figura 3. Ejemplo de especificación de atributos.

La segunda situación en la que un atributo con múltiples perspectivas es utilizado, es cuando es representado en diferentes perspectivas de acuerdo con un contexto en el cual el objeto es mostrado. En esta situación, no existe atributo *default*.

Algunos ejemplos:

nombre: String

título: String = Conferencia Nacional

gratificación: Real = 0 {oriunda da circular 001}

descripción: [Texto+, Imagen]

descripción: [Texto+, Imagen = desc1.bmp]

descripción: [TextoMarketing, TextoInvestigación]

Operaciones

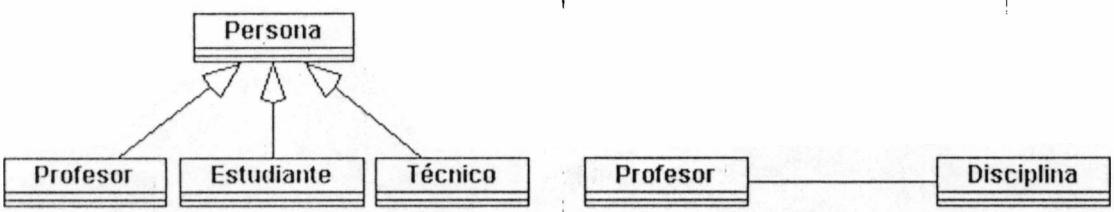
Cuándo es necesario, las clases pueden presentar operaciones, es decir, pueden presentar ejecutar sobre las informaciones de la aplicación.

Ejemplo:

Visibilidad nombre (lista-parámetros): expresión-resultado {propiedad}

Relaciones

Las relaciones (denominadas "Asociaciones" en UML) representan los lazos entre objetos. Al igual que en UML, las relaciones pueden ser de especialización (is-a), asociación, composición, y pueden poseer cardinalidad.



Figuras 4 y 5. Ejemplos de relaciones es-un y una relación binaria.

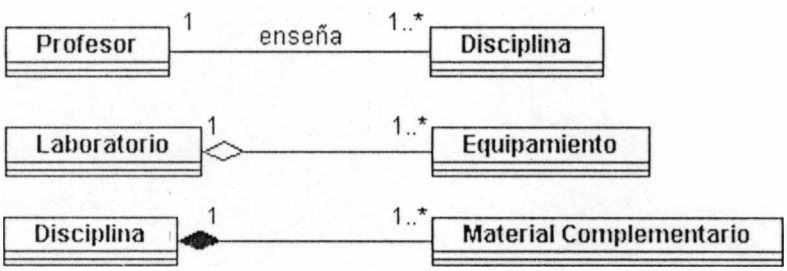


Figura 6. Varios ejemplos de relaciones con cardinalidades, agregación y composición.

Subsistemas

Los subsistemas representan un agrupamiento de clases y relaciones, que tratan un mismo tema y, juntos, son independientes del resto de la aplicación, es decir, son abstracciones de un sistema conceptual completo. Los subsistemas en OOHDM poseen una semántica semejante a los packages de UML.

2.3 Modelo Navegacional

El modelado navegacional define ls informaciones que serán presentadas, y las posibles formas de navegación entre ellas. El modelo navegacional es definido a partir de un modelo conceptual. Para un mismo modelo conceptual, pueden ser definidos uno o más modelos navegacionales. En este último caso, para un mismo dominio son definidas diferentes aplicaciones, es decir, para cada perfil de usuario son definidas diferentes aplicaciones conteniendo parte o toda la información contenida en el esquema conceptual, de acuerdo a las necesidades del usuario.

En el modelo navegacional son especificados cuales serán los objetos navegacionales (nodos, contextos, estructuras de acceso, etc.) que serán vistos por los usuarios.

El modelado conceptual produce las siguientes salidas:

a) Un esquema de clases navegacionales conteniendo la definición de los nodos y los links.

b) Un esquema de contextos identificando los contextos navegacionales y las estructuras de acceso.

c) Un esquema de clases en-contexto.

Hipermedia

Consideramos hipermedia como la extensión multimedia al hipertexto. Definimos hipertexto como una red de nodos y links que permite una estructuración no-lineal de la información, y la posibilidad de navegar a través de la red con o sin un orden predeterminado, según la necesidad del usuario. "

Nodo

Los nodos son piezas de información auto contenida. Los nodos pueden categorizarse de acuerdo a su estructura interna en:

Nodos Atómicos: Son los componentes más primitivos de esta capa.

Nodos Compuestos: Están formados por otros nodos arreglados en una jerarquía de composición, o "parte-de". Esto introduce una semántica alternativa de organización de la información y de navegación a través de la misma.

NODE *nombre* **[FROM** *nombreClase*: varName] **[INHERITS FROM** *claseNodo*]

attri: type1 **[SELECT** nombre1] **[FROM** class1:varName1, classj: varNamej

WHERE logical expression]

attr2: type2 **[SELECT** nombre2]...

...,

attrn: typen [idem]

END

Donde:

nombre: es el nombre de la Clase de Nodo que estamos creando.

className: e nombre de la Clase Conceptual (de donde el nodo está siendo mapeado).

nodeClass: nombre de la superclase.

attri: son los nombres de atributos para esa clase, type1: el tipo del atributo.

nombrei: son los sujetos para la expresión del query y vari son variables utilizadas para expresar condiciones lógicas.

expresión lógica: permite definer clases cuyas instancias sean combinaciones de objetos definidos en el esquema conceptual cuando ciertas condiciones se cumplan en sus atributos y/o relaciones.

Con el fin de definir a los nodos como *views* o "vistas" orientadas a objetos de las clases conceptuales definidas durante el modelo conceptual, utilizamos un lenguaje de consulta [Kim90]. Se permite además que un nodo sea definido mediante la combinación de diferentes clases relacionadas en el esquema conceptual.

Link

Los links representan la relación entre dos o más nodos. Conforman los arcos de la red hipermedial. La característica distintiva de un link es que puede ser "navegado", es decir, ser activado para pasar de un nodo a otro/s que están relacionados mediante ese link. Es así como el usuario irá formando el recorrido deseado con o sin un orden predefinido.

Los links son accedidos a través de anclas contenidas en la información que el nodo presenta, o a través de una estructura de acceso. Hablaremos de esos otros componentes en sus ítems correspondientes. Por otro lado un link llega a un destino que en nuestro modelo será un nodo completo o un conjunto de nodos. Los links pueden tener atributos. Un ejemplo sería el significado de la relación.

Clase de nodo y clase de link

Simulando la forma en que las clases factorizan la estructura y comportamiento de sus instancias en el paradigma OO, se definen los conceptos de clase de nodo y clase de link. Una clase de nodo define las propiedades comunes de cierto grupo de nodos que contienen los mismos atributos (aunque cada uno tendrá un valor distinto para los mismos), se muestran de la misma forma y se relacionan con otros nodos de igual manera.

Una clase de link agrupa aquellos cuyo origen pertenece a la misma clase de nodos, al igual que su destino. Si consideramos atributos en los links, aquellos pertenecientes a la misma clase mantienen los mismos atributos.

Link Unarios y Binarios

Del mismo modo que las relaciones en un modelo conceptual, los links pueden establecer diferentes características, como por ejemplo la cardinalidad.

Un link puede conectar 2 instancisa de nodos diferentes(binario) o de un mismo nodo (unario).

En el esquema navegacional, solamente las relaciones conceptuales que permiten navegación entre objetos son representados como links. Su representación se realizada de modo semejante a las relaciones en el modelo conceptual.

Link Unidireccional y Bidireccional

Los links unidireccionales representan que la navegación entre los objetos solamente es permitida en la dirección de la flecha. Como es de esperar, los links bidireccionales permiten la navegación en ambos sentidos.

Cardinalidad de Links

Un atributo de clase navegacional que representa un link con cardinalidad 1..N debe ser de tipo ***Índice*** o un link a un índice, en tanto que el atributo de la clase navegacional que prepresenta un link con cardinalidad 1 debe ser de tipo ***Anchor***.

Ordenación

Cuando el papel de un Link representa una cardinalidad mayor que 1, los elementos pueden estar ordenados o no. Por default los elementos no están ordenados y, en este caso, no se necesita especificar nada mas. Mientrs tanto, en el nivel navegacional, no se especifica como será hecha la ordenación ni como será mantenida.

Agregación y Composición

La Agregación y la Composición del Modelado Navegacional posee la misma semántica del Modelado Conceptual. Una Agregación indica que un objeto esta compuesto por otros objetos que no son dependientes de este.

Diagrama de Objetos

El diagrama de objetos del modelado Navegacional tiene la misma función de un diagrama de objetos en el modelado conceptual, donde se muestra un conjunto de instancias relacionadas, de acuerdo con el diagrama de clases navegacionales definido.

Contextos

Un contexto de navegación es un conjunto de objetos (nodos, links y recursivamente otros contextos) que están relacionados de acuerdo con algún aspecto (e.g. los nodos de una misma clase que presentan un atributo común o mismo contenido, los nodos del mismo tipo que presentan una relación con un mismo nodo de otro tipo, los nodos de tipos diferentes que presentan una característica común). Los objetos contenidos en un contexto pueden ser de clases diferentes. Como ejemplo de contextos navegacionales podemos citar: todos los profesores de un departamento, los profesores de una determinada área de investigación, las disciplinas dictadas por un profesor, los profesores y estudiantes que practican un determinado deporte, etc.

Los objetos de un contexto de navegación pueden contener varios contextos anidados. Por ejemplo, un profesor puede participar del contexto Profesor por departamento (donde son accedidos todos los profesores) y del contexto de Profesor por Área (donde son accedidos todos los profesores de una misma área). Vale notar que independientemente del contexto en el que un objeto es accedido, este presenta todos los atributos definidos en el esquema de navegación. No obstante, de ser necesario un nodo puede presentar atributos diferentes de acuerdo al contexto en el cual esta siendo accedido, si ello fuera necesario, tal como veremos mas adelante.

Contextos Enumerados

Los elementos de un contexto de navegación pueden ser escogidos arbitrariamente a partir de una o mas clases (contexto enumerado). Como ejemplo, de este tipo de contexto de navegación podemos citar los "Tours Guiados" disponibles en varias aplicaciones hipermediales en los cuales se visitan los nodos mas representativos de la hipermedia.

Contexto Por Consulta

Un contexto de navegación también puede estar formado por los elementos resultantes de una consulta realizada en tiempo de ejecución. Los parámetros de esta consulta no pueden ser previstos en el sistema antes de la ejecución de la propia consulta. Estos contextos son definidos en ejecución, automáticamente por el sistema o interactivamente por el usuario.

Grupo de Contexto

En OOHDM, además del concepto de contexto de navegación, existe el concepto de grupo de contexto, que es un conjunto de contextos. La ventaja de poder especificar un grupo de contextos es la compactación de la notación, a través de una especificación parametrizada.

Parámetro de Link de Acceso a un Contexto

El link de acceso a un contexto pude representar un parámetro. En este caso, el parámetro puede indicar un elemento de un contexto, un elemento de un grupo de contexto o un contexto en un grupo de contexto.

Generalización de Contexto

Cuando un esquema navegacional presenta una generalización de clases navegacionales, el esquema de contextos también puede presentar una generalización de contextos.

La generalización de contextos es utilizada para representar que todas las subclases de una clase navegacional presentan uno o más contextos en común. Estos contextos son incluidos como contextos de superclase y heredados por todas las subclases.

Contexto Persistente y No-Persistente

Los contexto de navegación pueden ser persistentes o no. Un contexto no persistente existe solamente durante una sesión de navegación donde fue creado. Un contexto persistente en cambio, luede de ser creado, puede ser accedido en sesiones posteriores.

Contexto Dinámico

Un contexto puede ser dinámico o no. Un contexto de navegación dinámico es aquel cuyos elementos son definidos o alterados en tiempo de ejecución.

Navegación de los Contextos

La navegación dentro de un contexto pede ser realizada de diversas maneras:

Navegación secuencial: los elementos del contexto son accedidos en forma secuencial y pre-establecida. La navegación secuencial del contexto define un primer elemento, un último elemento y el próximo y anterior de cada elemento.

Navegación Circular: Los elementos del contexto también son accedidos en orden secuencial y preestablecida, y pueden circular. Se distinguen de la navegación secuencial en que el primer y último elemento no están definidos.

Navegación Libre: Los elementos del contexto no precisan ser accedidos en orden. En esta navegación, los elementos pueden ser accedidos desde cualquier otro.

Navegación por Índice: Los elementos de un contexto son accedidos solamente a partir de un índice, o sea, luego de acceder un elemento a través de un índice, obligatoriamente se retorna al índice, y a partir de este es accedido e próximo elemento deseado.

Combinación de navegación por índice con secuencial o circular: los elementos del contexto pueden ser accedidos tanto por un índice como por el elemento anterior o próximo.

Destino de link

El destino del link puede ser singular o múltiple. En el primer caso el destino contiene al nodo en cuestión, más cierta información referida a la forma en que va a ser presentado de acuerdo al link por el que se llega a él (veremos esto más adelante con el concepto de vista de nodo). En el segundo caso en lugar de un nodo, el destino contiene a un conjunto de nodos.

La forma de mostrar un destino múltiple coincide en la mayoría de los sistemas en proveer una estructura de acceso intermedia para seleccionar un nodo particular, aunque los modelos no lo especifican por considerarse una decisión de implementación.

2.4 Diseño de Interfaz Abstracta

Esta actividad comprende la construcción de un modelo abstracto de la interfaz que hará perceptible el esquema de navegación definido en el paso anterior. Esto implica definir los objetos de interfaz que el usuario percibirá, y la forma particular en la que los nodos, links y estructuras de acceso se van a mostrar, qué objetos de interfaz activarán la navegación, la forma en la que los elementos multimediales serán sincronizados, y las transformaciones de interfaz que tendrán lugar.

Nuevamente, la separación de esta etapa permite construir diferentes interfases para el mismo modelo de navegación. OOHDM propone la utilización de un modelo formal denominado Abstract Data Views, para describir la interfaz de los objetos de navegación.

2.5 Implementación

En esta actividad el diseñador debe realizar un mapeo de los modelos de navegación y de interfaz abstracta en objetos concretos, disponibles en el ambiente elegido de implementación.

Más allá de las actividades en las que se separa esta metodología, los principales conceptos que define son: Nodo, Clase de Nodo, Link, Clase de Link, Ancla, Contexto de navegación, Estructura de Acceso.

3 Una Herramienta para modelar con OOHDM.

3.1 Introducción

Esta herramienta fue desarrollada con el objetivo de proveer un medio para expresar diseños con la notación OOHDM, utilizando como medio de persistencia una base de datos orientada a objetos (Poet).

Para la construcción del editor gráfico, se utilizó el framework JHotDraw como base para desarrollar los diferentes editores y notaciones correspondientes a las diferentes actividades establecidas por la metodología. Sin embargo JHotdraw por si solo apenas provee los elementos de base y con una restricción muy importante: solo soporta en forma nativa las capacidades graficas del framework de interfaz original de Java, el AWT. Esto se debe al hecho de que, si bien esta montado sobre una interfaz mas amigable como resulta Java Swing, en sus clases básicas la tarea de "pintado" de la interfaz se realiza sobre la base de las capacidades de `java.awt.Graphics` que provee herramientas insuficientes para implementar las diferentes notaciones de OOHDM (e.g. no es posible dibujar un trazo punteado, etc.).

Por tal motivo se realizo un estudio detallado del framework, sus hotspots, arquitectura e implementación, y se agregaron modificaciones de tal manera que las aplicaciones existentes no se vieran afectadas (los ejemplos estándar del framework siguen funcionando con el framework modificado) y adoptar el criterio arquitectural que tenía el framework hasta ese momento.

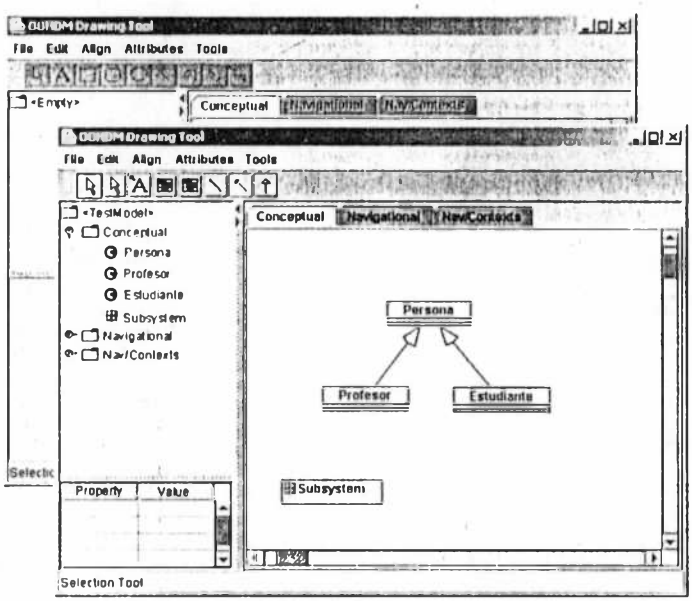


Figura 7. Dos imágenes de la herramienta de modelado con OOHDM.

A lo largo del desarrollo, se divisó la necesidad de desarrollar diferentes herramientas sencillas para levar a cabo tareas como por ejemplo incorporar una clase desde una aplicación embebida en un .jar y mapearla a una clase conceptual, o simplificar la creación de clases navegacionales a partir de las conceptuales.

Una implementación posible para estas herramientas es a través de wizzards de interfaz que guíen al usuario en la tarea. Por desgracia, los frameworks de interfaz gráfica como Swing no soportan este tipo de abstracciones, sino que proveen los elementos básicos como son las ventanas de Diálogo (modales) y los diversos controles o widgets de interfaz (caso similar al de JHotDraw vs. el trabajo a realizar descripto anteriormente). Es por este motivo que se decidió diseñar un mini framework que simplifique la implementación de estas herramientas, abstrayendo al desarrollador de la implemenación de la lógica de controls y manejo de interfaz. Como resultado colateral, se obtienen herramientas con una apariencia mas uniforme.

3.2 El Framework JHotDraw

JHotDraw define un esqueleto básico para un editor de GUI-based con las herramientas en una gama de colores de la herramienta, diversas visiones, figuras gráficas definidas por el usuario, y una ayuda para el ahorro, el cargamento, e imprimir gráficos. El marco se puede modificar para requisitos particulares usando herencia y combinando componentes.

JHotDraw es interesante desde un punto de vista de la tecnología de dotación lógica también. Convertido originalmente en palique por la cuba de tintura y la sala Cunningham de Kent, JHotDraw era uno de los primeros proyectos del desarrollo del software lógica diseñados explícitamente para la reutilización y etiquetó un marco. También fue documentado muy temprano en términos de los modelos del diseño

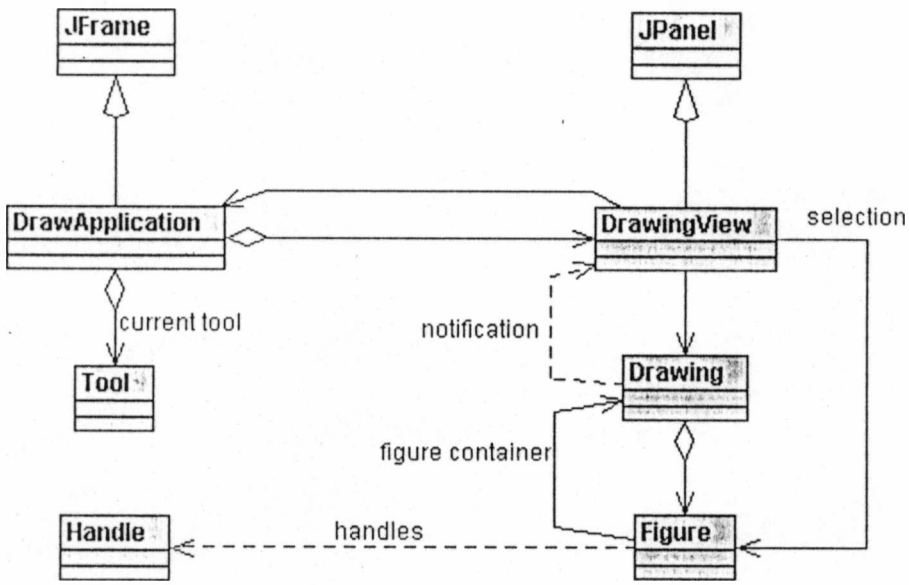


Figura 8. Diagrama general del Framework JHotDraw.

Todas las clases y los interfaces de JHotDraw se ordenan en conjuntos según sus funciones. El conjunto `ch.ifa.draw.frramework` contiene sobre todo definiciones de interfaz de los requisitos componentes de la base -- su responsabilidad, funciones, e interoperation.

Cualquier aplicación que utilice JHotDraw tiene una ventana (Window) dedicado para trazar. Este DrawWindow es el Window del editor y es una subclase de javax.swing.JFrame. Contiene unos o más marcos internos, cada uno asociado a una visión que traza. El DrawingView, una subclase de javax.swing.JPanel, es un área que puede visualizar un gráfico y valida la entrada de información del utilizador. Los cambios en el gráfico se propagan al DrawingView que es responsable de poner al día cualquier gráfico.

El dibujo consiste en las figuras, que alternadamente pueden ser contenedores para otras figuras. Cada figura tiene manijas, que definen puntos de acceso y se determinan cómo obrar recíprocamente con la figura (por ejemplo, cómo conectar la figura con otra figura en un DrawingView, usted puede seleccionar varias figuras y manipularlas. El DrawWindow sí mismo tiene generalmente una herramienta activa de la gama de colores de la herramienta, que funciona encendido el dibujo asociado al DrawingView actual.

3.3 Mini Framework para desarrollar Wizzards en Swing.

Como anteriormente describimos en la sección 3.1, se ha desarrollado un mini framework para la implementación de wizzards.

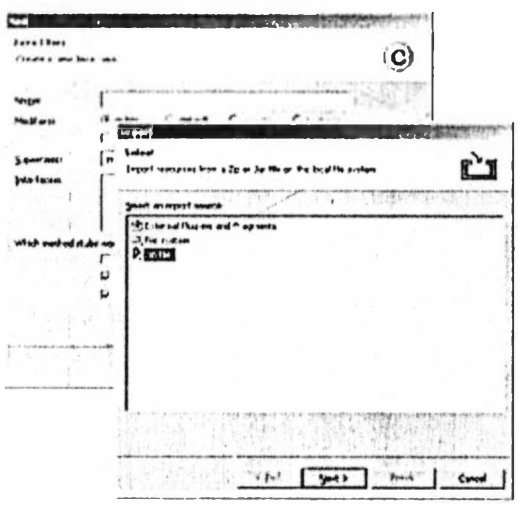


Figura 9. Dos ejemplos de wizzards construidos con el framework.

Los componentes framework se definen como:

-Una interface Wizzard describe la funcionalidad básica de un wizzard.

-Una clase abstracta `WizardFrame` que impleenta la interfaz Wizzard de ejecución y lo hereda de `javax.swing.JFrame`.

-Una interfaz `WizardComponent` que describe paneles, los botones, y los otros objetos que se pueden agregar a un mago básico.

-Varias implementaciones de paneles y de los botones basados en WizardComponent que se puede agregar a un mago básico.

-Una clase concreta de `WizardData` que contiene toda la información utilizó en un mago.

-Una excepción WizardException ampliada de java.lang.Exception

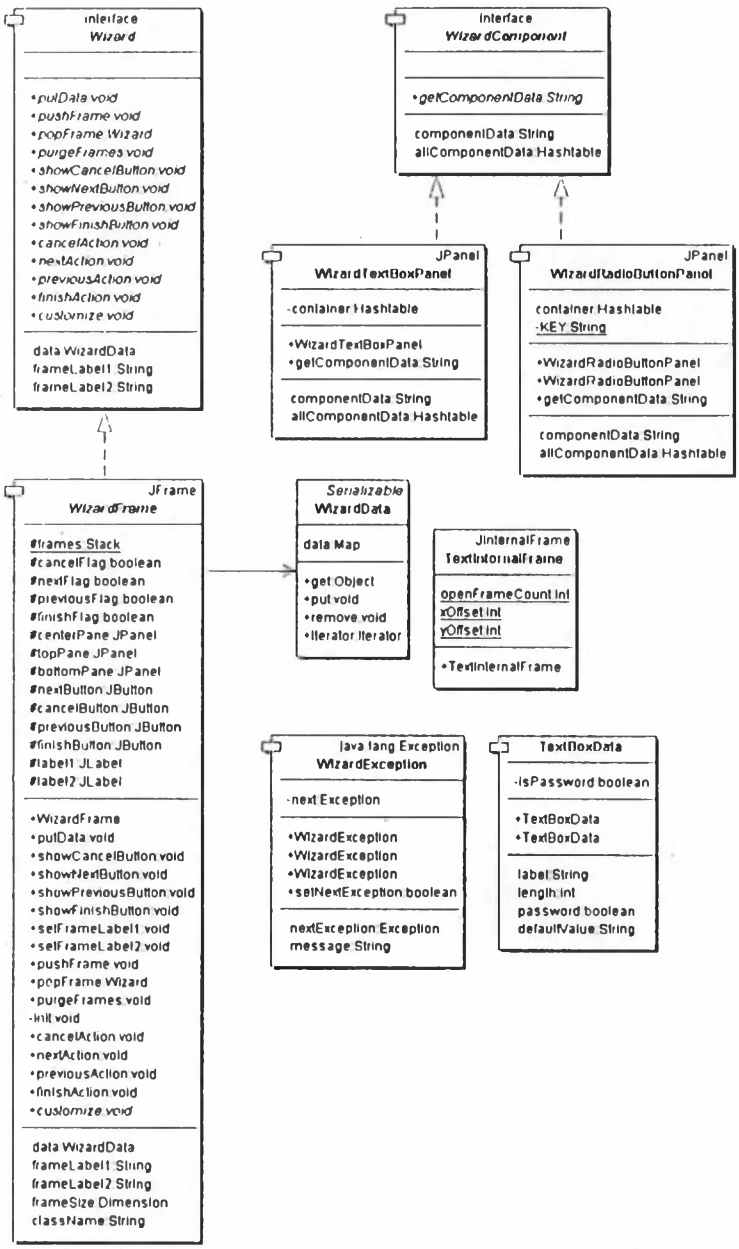


Figura 10. Diagrama de clases del mini-framework para el desarrollo de Wizzards.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

4 Soporte para Hipermedia en Web

4.1 Tecnologías de generación de contenidos en WWW.

Arquitecturas para Clientes "finos" (Thin Clients)

Desde la aparición de las tecnologías de COM y Javabeans, las técnicas de generación de presentación se han basado en la utilización de componentes especializados que corren en el servidor y generando de esa forma el middle-tier o capa intermedia.

Si bien la eficacia de la utilización componentes de software para la resolución e implementación de reglas de negocio en el middle-tier no es discutida, el debate se centra en cuál es la mejor forma de resolver la presentación. Entre los problemas que habitualmente se discuten se encuentran:

La separación de la presentación (HTML) de la generación de contenidos (código que accede a bases de datos, etc.).

Mantenimiento de consistencia con un paradigma moderno de construcción basado en componentes para maximizar la reusabilidad y modularización.

Reusabilidad y anidamiento de tags en diferentes contextos de páginas (no en el sentido de contexto de OOHDM) en forma de templates.

Integración de modelos sofisticados al web.

Minimizar la curva de aprendizaje de nuevos lenguajes de scripts y conjuntos de tags especializados, y maximizar la compatibilidad con las herramientas de edición de HTML.

La combinación de código y presentación oscurece el diseño y fuerza al diseñador y programador al diseño de la página y el software simultáneamente, aún tratándose de actividades muy diferentes. Ese escenario se agrava cuando se intenta que ambos trabajen separadamente, puesto que en muchos casos resulta inevitable el conflicto de los cambios mutuos de código y programación.

El descuido por la construcción de componentes orientado a objetos y la imposibilidad de reutilización y el anidamiento de bloques se encuentran íntimamente relacionados con la fusión de código y HTML. Esta mezcla significa que la generación de código se halla acoplada a un formateo específico del HTML.

Describimos a continuación las diferentes tecnologías disponibles actualmente, que han sido desarrolladas en un intento por superar los problemas anteriormente descriptos. Estas técnicas desarrolladas mejoran los documentos HTML empleando lenguajes de scripting, conjuntos de tags especializados y generación automática de clases como una ayuda para resolver los primeros 3 problemas enumerados.

Componentes-servidores (servlets)

Llamaremos indistintamente servlets a los componentes COM server-based y los Java servlets propiamente dichos, ya que conceptualmente son muy similares y no se hace referencia a capacidades exclusivas de estas tecnologías.

Los servlets constituyen un notorio mejoramiento respecto del modelo CGI que implicaban un esquema muy pesado de al menos un proceso dedicado por pedido. El modelo de componentes en el servidor maneja los pedidos (requests) utilizando threads livianos –también bajo la premisa de uno por pedido– que son mucho más eficientes en términos de consumo de recursos del servidor. Estos mini-programas agregan también un grado de persistencia a través de los pedidos sin involucrar accesos a una RDBMS o almacenamiento a disco.

No obstante, el cambio de CGI por Servlets no soluciona el problema del HTML profundamente embebido en el código (sea Perl, C++, Delphi o Java) que genera el contenido de la página. Los servlets comúnmente son utilizados en la generación dinámica de páginas agregándoles el texto HTML y dejando que el servlet lo comunique como su salida haciendo `java.io.PrintWriterobject` o `ASP.response.write` en el caso de COM/ASP. Este tedioso procedimiento acopla la generación de presentación y el contenido. Como resultado, o bien el diseñador aprende COM/JAVA o el programador debe aprender diseño de páginas.

La construcción de componentes que se ocupen de la presentación es viable si se incluye algún mecanismo para utilizar y reutilizar los bloques de código HTML dentro de los cuales el contenido dinámico pueda ser embebido el momento de resolver un requerimiento. El resto de las técnicas se basan en la adición de alguna forma de template a los servlets crudos.

4.2 Mecanismos de templates

El uso de templates es una técnica que mejora los documentos estáticos de HTML agregándoles algún código de scripting o mediante la automatización de creación de componentes que representen a los documentos, como una forma de "compilación" de los mismos. Un template es entonces, una pieza de código HTML extendido que mediante la manipulación y compilación, o mediante el agregado de scripting puede facilitar la inclusión de los contenidos generados dinámicamente. El resto de esta sección describe brevemente tres esquemas de templates actualmente disponibles (+Delphi).

Java Server Pages (JSP)

La tecnología de las páginas de JavaServer (JSP) proporciona una manera simplificada y rápida de crear páginas del Web que visualicen contenido dinámicamente generado. La especificación de JSP, desarrollada como una iniciativa a nivel industrial conducida por Sun Microsystems[Sun_JSP], define la interacción entre el servidor y la página de JSP, y describe el formato y el sintaxis de la página.

Las páginas de JSP utilizan tags XML y los scriptlets escritos en el lenguaje de programación de Java encapsulan la lógica que genera el contenido para la página. Transforma cualquier tag del formato HTML o XML directamente de nuevo a la página de la respuesta. De esta manera, las páginas de JSP separan la lógica de la página de su diseño y visualización. La tecnología de JSP es parte de la familia de la

tecnología Java; utiliza un lenguaje de "scripting" Java, y las páginas de JSP se compilan en servlets.

Las páginas de JSP pueden llamar a componentes JavaBeans o Enterprise JavaBeans para realizar un proceso en el servidor. Como tal, la tecnología de JSP es un componente dominante en una configuración altamente escalable para las aplicaciones Web. Las páginas de JSP no se restringen a ninguna plataforma o webserver específico

En JSP las páginas se compilan en servlets, de modo se podría escribir teóricamente servlets para toda la aplicación Web. Sin embargo, la tecnología de JSP fue diseñada para simplificar el proceso de crear las páginas separando la presentación de Web del contenido. En muchas aplicaciones, la respuesta enviada al cliente es una combinación de los datos del modelo y de los datos generados dinámicamente. En este caso, es mucho más fácil trabajar con las paginas JSP que hacer todo con servlets.

Apache Velocity

Velocity[Velocity] provee un framework minimalista para implementar un esquema *Model/View/Controller* (MVC) en el contexto del Web. Este framework facilita al mapeo de los pedidos y la ejecución de los templates, y les provee la información de la sesión y cualquier business object que necesite acceder. El lenguaje de script permite que los templates accedan en forma aleatoria a los business objects. Haciendo una semejanza con el MVC, los business objects comprenden el *modelo*, los templates son las *vistas*, y el código que mapea los pedidos y ejecutan los templates es el *controlador*.

La característica distintiva de Velocity es el soporte de un lenguaje de scripting sencillo muy similar al Perl, que soporta:

-Iteración sobre listas de objetos.

-Acceso datos generados por objetos Java externos.

-Composición (vía inclusión) de otros templates y de cualquier otro archivo de texto.

-Parseo de los archivos template (con lo cual la inclusión es dinámica).

-Variables locales.

Para el acceso de los objetos Java externos, Velocity utiliza ampliamente las capacidades de reflexión provistas por Java durante el proceso para establecer las interfases válidas de invocación. Esto permite a los templates acceder a datos sin necesidad de una sintaxis especial para declaración de objetos, el binding de identificadores locales a clases de objetos y sus propiedades, o proveer getters y setters.

FreeMarker

Es muy similar al Velocity, e implementa una arquitectura MVC tal cual fue descripta para Velocity. Las principales diferencias son:

Un lenguaje más sofisticado (posee funciones y expresiones boolcanas).

Sintaxis más compleja.

Mecanismo de caching de templates más sofisticado.

XMLC

El concepto que implementa esta herramienta es completamente diferente a los ya descriptos. XMLC es un compilador Java especializado para HTML (o XML), cuyo resultado es la generación de una clase por cada archivo HTML. La idea es condensar en una clase la estructura DOM de representación del documento en cuestión. Las clases proveen métodos para acceder a los elementos del HTML con atributos ID, y el código Java manipula los documentos utilizando dichos métodos para modificar el template agregando o cambiando los correspondientes elementos. XMLC utiliza el elemento SPAN para la inserción y modificación de texto, y el atributo CLASS para agrupar conjuntos de elementos.

La idea de fondo de esta tecnología es que las páginas HTML permanezcan como un esqueleto válido del website, permitiendo que el diseñador continúe la evolución de las páginas y los cambios puedan ser transmitidos al programador sin perder la información necesaria para la generación dinámica de la página.

4.3 Modelos de Arquitecturas de Aplicaciones Web.

La literatura en tecnología de aplicaciones Web se utiliza con frecuencia el término "*modelo 1*" y "*modelo 2*" (MVC) sin la explicación. Esta terminología proviene de las especificaciones iniciales de JSP, que describió dos modelos básicos del uso para las páginas de JSP. Mientras que los términos han desaparecido del documento de especificación, los términos continúan en uso. El modelo 1 y el modelo 2 refieren simplemente a la ausencia o a la presencia de esquema *Model-View-Controller* que maneje las acciones del usuario.

4.3.1 Arquitectura "Modelo 1"

La arquitectura "*modelo 1*" consiste en un web browser que tiene acceso directo a las páginas de JSP. Las páginas JSP acceden a los JavaBeans que realizan operaciones de negocio en el modelo de los datos, y la página siguiente (página de JSP, servlet, HTML PAGE, los etc.) a visualizar se determina por los links seleccionados en el documento fuente, o por los parámetros pasados en las operaciones GET y POST. El controlador en el *modelo 1* es descentralizado, puesto que cada página JSP o servlet procesa su propia entrada de información (los parámetros de GET y POST), y la siguiente página se determina por la página actualmente accedida.

En esta clase de arquitectura de aplicaciones, la información del flujo entre las operaciones y las *views* generalmente queda embebida en las mismas *views* (en este caso las páginas JSP).

La presencia de directivas tales como `RequestDispatcher.forward()` en una página JSP implica que hay lógica embebida en la pagina que decide la pagina siguiente a ser visualizada. Luego de un tiempo, como la aplicación crece, los cambios en la lógica del workflow se acumulan. La aplicación llega a ser difícil de mantener porque el workflow se distribuye a través de las múltiples views (páginas JSP).

4.3.2 Arquitectura "Modelo 2"

La arquitectura denominada "*modelo 2*" introduce un controlador entre el browser y las páginas JSP o el contenido del servlet que son entregados. El controlador centraliza la lógica para enviar peticiones a la siguiente vista, basada en la URL de la petición, la información de los parámetros, y el estado de la aplicación. La arquitectura "*modelo 2*" desacopla las paginas JSP y los servlets entre si, y pone la responsabilidad del flujo de la página dentro del Controller. El patrón de diseño "J2EE Front Controller"[J2EE] describe las ventajas de centralizar toda la petición enviada por la aplicación en un solo servlet. El controlador es también una aplicación del patrón "Mediator", puesto que desacopla las views entre si.

Las aplicaciones con una arquitectura MVC son más fácil de mantener y de extender, puesto que las views no se referencian directamente entre si. El Controller del modelo MVC proporciona un solo punto de control para la seguridad y el logueo, y habitualmente encapsula los datos entrantes en una forma utilizable por el modelo. Mientras que la configuración del MVC agrega una cierta complejidad a la aplicación, un framework de aplicación MVC puede simplificar en gran medida esta tarea. Los frameworks MVC tales como Apache Struts, JavaServer Faces y SourgeForce Maverick incluyen como controller un servlet configurable, y proporcionan un conjunto de clases abstractas que pueden ser extendidas para manejar las peticiones de la aplicación. Algunos frameworks incluyen lenguajes macro u otras herramientas que facilitan la implementación de las aplicaciones MVC.

Los beneficios en la adopción de una arquitectura MVC en aplicaciones web son múltiples:

Desacopla la lógica y la presentación en componentes separados. Los frameworks alientan la separación de la presentación y de la lógica porque la separación se diseña en las interfases de extensión.

Separa roles del desarrollador. Los frameworks de aplicación proporcionan generalmente diversas interfases para diversos desarrolladores. Los desarrolladores de componentes de presentación tienden a focalizarse en crear las pagina JSP usando custom tags, mientras que los desarrolladores de la lógica tienden para escribir clases , tag handlers, y el modelo. Esta separación permite que ambos tipos de desarrolladores trabajen más independientemente.

Proporciona a un punto central de control de control. Los servlets controller en la mayoría de los frameworks tienen un amplio conjunto customizable de características templating, internacionalización, control de acceso, y registración.

Facilita la prueba de unidad. Puesto que las interfases componentes del framework son consistentes, es posible la utilización de herramientas de testing automatizadas.

Estabilidad. Un framework creado y mantenido activamente por una base amplia de usuarios, habitualmente resulta ser más estable que algo creado ad-hoc.

Puede simplificar la internacionalización. La mayoría de los frameworks utilizan una estrategia flexible de la internacionalización.

Puede simplificar la validación de la entrada de información del usuario. Muchos frameworks tienen maneras consistentes para especificar la validación de la entrada de información. La validación es comúnmente disponible en el cliente, en el servidor, o ambas.

Los frameworks MVC son una capa de software lógica entre los interfases de la aplicación y de la capa Web que simplifican la creación de aplicaciones Web. En la Figura 11 vemos a la "pila" de servicios que forman parte de una aplicación. La plataforma de J2SE proporciona la plataforma de Java sobre el sistema operativo y el contenedor Web proporciona a los servicios relacionados con las peticiones de Web. Un framework de la Capa Web MVC, integrado por clases estructurales, maneja los detalles de las operaciones y el enviando de dichas peticiones.

Un framework MVC hace que las tecnologías de la capa Web sean más fáciles utilizar, ayudando a los desarrolladores a concentrarse en la lógica del negocio. Varios frameworks de MVC para la capa Web han aparecido desde hace un tiempo. Su aplicación aparece también en las recomendaciones de arquitecturas con tecnología J2EE.

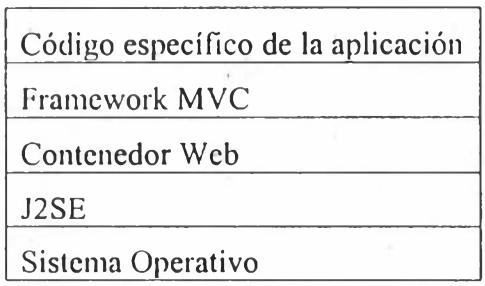


Figura 11

4.3.3 Aspectos de implementación de la Arquitectura MVC en aplicaciones web.

Existe una serie de aspectos a considerar en el desarrollo de aplicaciones con una arquitectura MVC. En la Figura 12 vemos un esquema general de una implementación de la arquitectura MVC en aplicaciones web.

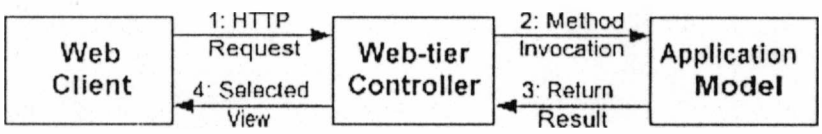


Figura 12. Digrama con el controller mediando entre el cliente web y el modelo de la aplicación.

Las aplicaciones con un solo tipo de cliente pueden implementar un solo tipo de controller, tal como aparece en la Figura 12. El controller recibe los requests HTTP, traduce los contenidos de éstos en operaciones sobre el modelo de la aplicación y devuelve una vista HTML o XML con los resultados. Los clientes futuros pueden ser acomodados mediante la creación de un controller por cada tipo de cliente, como vemos en la Figura 13.

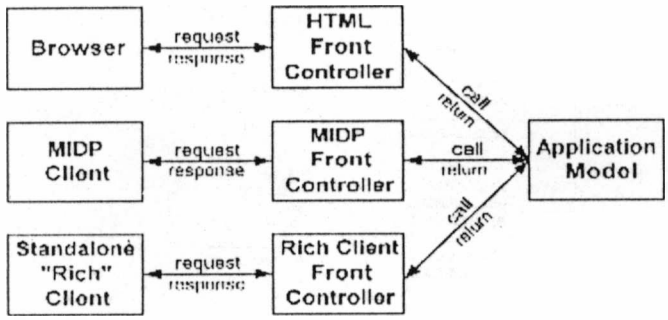


Figura 13. Arquitectura MVC soportando mútiples clientes con múltiples controllers.

Cierta funcionalidad de la aplicación, como por ejemplo la seguridad, puede resultar mas sencilla de administrar desde un solo punto. La introducción de un router de protocolo tal como aparece en la Figura 14, provee un solo punto de control para todos los clientes, y a su vez cada uno de ellos mantiene sus propios controllers.

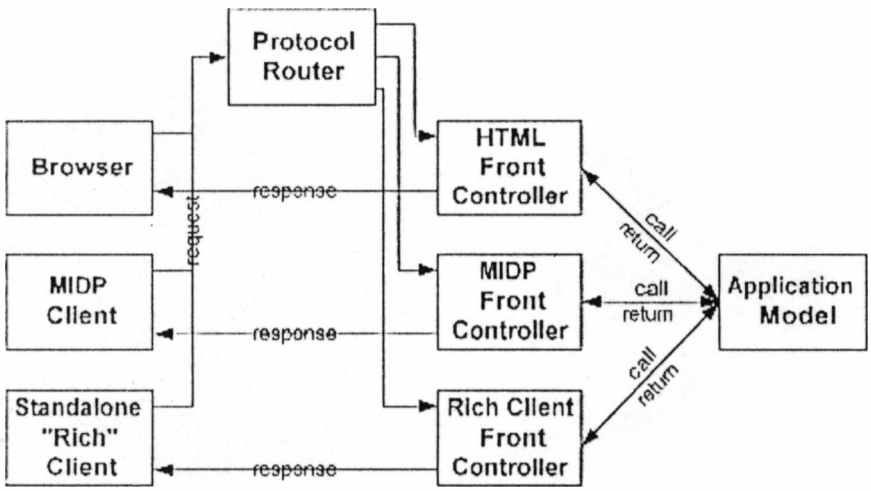


Figura 14. Uso de un router de protocolo para un control centralizado.

El router de protocolo puede ser un servlet o un servlet filtro, el cual determina el tipo del cliente basado el request header de HTTP y redirecciona el pedido al controller apropiado.

4.3.4 Controlando el flujo de la aplicación

Desde el punto de vista del usuario, una aplicación Web aparece como una secuencia de views cuya navegación se ve afectada de acuerdo a las funciones que sean ejecutadas. Determinar este flujo es la segunda gran responsabilidad del controller.

En esta sección, el término "*view*" corresponde en un sentido mas amplio a un recurso Web con una URL a partir de la cual este contenido esta disponible. Una view podría ser una página JSP, un servlet, o contenido estático, o alguna combinación de estos tres.

Generalmente, la "próxima" view a mostrar depende de varios factores:

- la view actual, indicada por la URL de pedido HTTP.
- los contenidos del pedido HTTP.
- los resultados de cualquier operación sobre e modelo de la aplicación, retornado por las invocaciones al modelo.
- posiblemente otras informaciones almacenadas en la sesión o estado de la aplicación.

El controller utiliza esta información para determinar cual view debe mostrar a continuación. Un controller Web "muestra" una view mediante el envío de la misma al cliente en algún formato que el cliente entienda (HTML, xml). El controller podría directamente producir el contenido y devolverlo, pero volveríamos a acoplar el controller con la view, y además resultaría muy difícil de extender.

En vez de que un controller se encargue de enviar la próxima view directamente, una mejor estrategia es delegar la creación de contenido a otros componentes de capa Web, sean servlets o páginas JSP. Un controller puede utilizar el pasaje del request para activar un servlet o JSP que produzca la próxima view. Esta estrategia simplifica en gran medida el controller, ya que lo único que tiene que realizar es determinar la URL de la próxima view y reenviar el request a dicha URL. EL componente en esa URL será el responsable de generar los contenidos que representan la próxima view.

Dado que determinar cuál view debe mostrarse a continuación es un proceso complejo que puede requerir acceder a datos de diferentes fuentes, es recomendable encapsular este proceso. De este modo, el controller delega la responsabilidad de determinar la próxima view a un administrador de flujo.

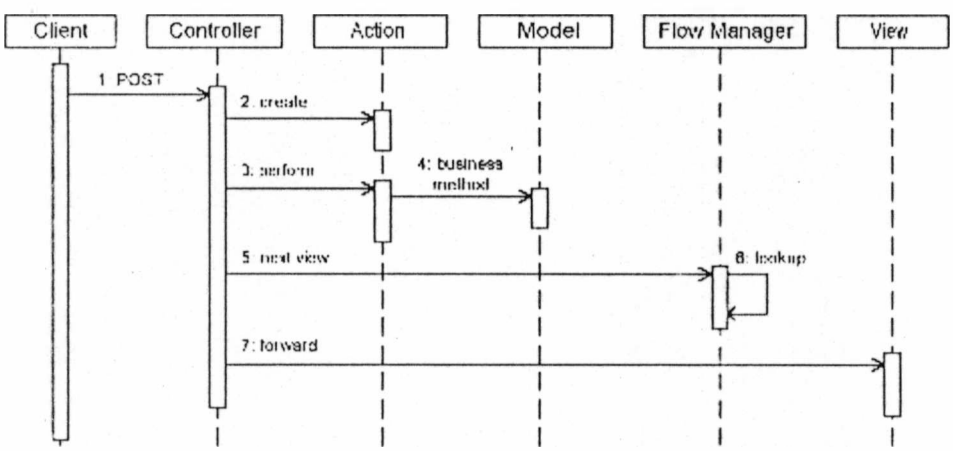


Figura 15. Diagrama de interacción del MVC para determinar la próxima view a navegar.

El controller [1] recibe un request del cliente, en [2] crea y [3] realiza la correspondiente Acción, la cual resulta en [4] llamados a métodos del modelo. Luego, [5] el controller invoca al flowManager (o "administrador de flujo"), el cual [6] selecciona la próxima view ser mostrada, y [7] reenvía ese pedido a la view seleccionada.

5 Arquitectura del Framework OOHDM

5.1 Introducción

Para documentar el diseño del framework se ha elegido presentar primeramente las jerarquías principales que forman su 'esqueleto', es decir, las abstracciones que lo delinean.

5.2 Diseño del framework

OOHDMModel

Es la clase a partir de de la cual derivan todos los elementos conceptuales mas importantes de un modelo OOHDM. Establece algunos aspectos que serán comunes a cualquier elemento de un modelo OOHDM, sea conceptual, navegacional, interfaz o proceso.

EL aspecto mas importante a destacar es la adopción del esquema planteado en el DynamicTemplate, donde la posibilidad de definir nuevas clases y establecer relaciones de herencia de forma uniforme a cualquier elemento de la metodología.

Aporta también la posibilidad de especificar restricciones de acceso (descripto por un objeto de la clase AccessPermission) y también una descripción textual que explique la existencia de determinado elemento de diseño, almacenado en una instancia de la clase Comment.

GeneralRelationship

Al igual que en el caso de OOHDMModel, esta clase define los aspectos comunes a cualquier relación de cualquier tipo presente en un diseño.

Subsystem

Modela la primitiva correspondiente a u subsistema en el diseño Conceptual, tal como se ha descripto en 2.2.

ConceptualClass

Modela la primitiva de clase conceptual (ver 2.2), y a pesar de ser un DynamicTemplate, no explota la posibilidad de generación de instancias, ya que su objetivo es establecer una representación del dominio de la aplicación subyacente, por lo tanto no tiene correspondencia con ninguna subclase de InstanceClass, es decir, la jerarquía de clases de objetos cuyas instancias tienen como tipo que las define a una instancia de algunas de las subclases de DynamicTemplate. Este tema será oportunamente desarrollado en la sección de patrones utilizados en este Framework (ver 5.3)

Nodo (Node)

Un nodo representa una pieza de información auto-contenida. Los nodos en una hipermedia son activados a través de links y se muestran en una determinada interfaz. A su vez desde un nodo se puede navegar hacia otros por intermedio de la activación de las anclas de links que posee.

La jerarquía encabezada por esta clase está basada en la estructura interna de los nodos, pero en forma totalmente independiente del tipo de información que muestren o el comportamiento que disparen. La Figura 16 muestra la jerarquía completa.

Un nodo puede mostrar distinta información o activar distinto comportamiento dependiendo del perfil del lector. Para lograr esto se permite que un nodo tenga distintas "vistas", una para cada contexto definido.

En su definición, OOHDM establece que un usuario siempre está navegando en algún contexto, aunque se trate del "contexto por *default*" es decir donde no se ha especificado una vista particular para la clase del nodo en cuestión. Esto implica que también existe siempre una vista (también por default) que funciona como un "decorador" transparente ya que a lo sumo hará un reenvío de los mensajes que reciba al nodo concreto. Entendemos por una view que decora a un nodo en el sentido de agregado transparente de atributos al nodo básico, relevantes al contexto en el cual se esta navegando.

Finalmente, existe una segunda implicancia en la noción de aumentación que realizan las vistas sobre los nodos que decoran, y es que esta era siempre aditiva. Por lo tanto el diseñador estaba forzado a pensar al momento de definir una clase de nodo navegacional como el factor común a todas las vistas que pudiera tener un nodo. Esto claramente es una restricción incómoda. En el modelo que aquí presentamos, esa restricción es eliminada y, como veremos mas adelante, una vista puede establecer que atributos (sean estos pertenecientes al nodo original o exclusivo de la vista) serán accesibles. De este modo, se implementa un mecanismo de visibilidad de atributos mucho más potente y flexible a los cambios y necesidades nuevas.

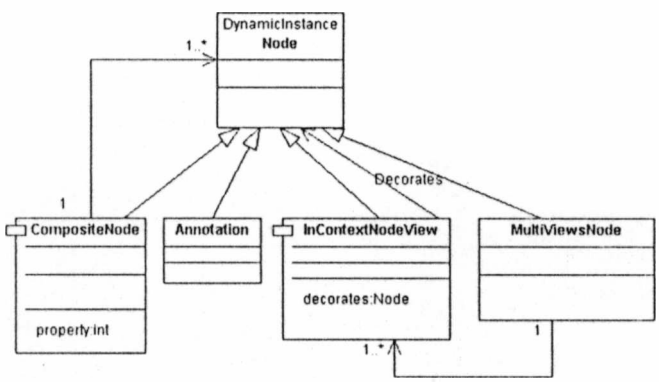


Figura 16. Jerarquía completa de Nodo.

Nodo de múltiples vistas (MultiViewsNode)

Representa los nodos que muestran un "cara" distinta según el contexto, o perfil de usuario actual. Es decir que estos nodos constituyen una agregación de "vistas de nodo".

Los objetos de esta clase agrupan objetos de la clase InContextNodeView .

Nodo asociado a objetos de la aplicación (Node)

Todos los Nodes responden a un "patrón" determinado por la clase del objeto que observan. Este patrón o tipo les dice los aspectos (mensajes) definidos en aquella clase que deberá utilizar para obtener y modificar los datos a mostrar y para disparar el comportamiento. Los colaboradores de los Nodes que hacen de "patrón" o "tipo" de los mismos están representados por la clase NodeClass. Los Nodes tienen que mantener una referencia explícita a su NodeClass. Como se dijo anteriormente, subclasificar Node por cada clase de cada aplicación a mapear no tendría sentido. Es por esto que se implementó un colaborador para la misma que simule a su clase dependiente del nivel de objetos. Este diseño responde al patrón Dynamic Template.

Como se dijo previamente, estos nodos también permiten el disparo de acciones a ser atendidas por los objetos de la aplicación. Este comportamiento del objeto a observar pueden ser especificado en el nodo mismo (con lo que podrá ser disparado desde todas las vistas) o en una vista particular.

La clase abstracta Node es subclasificada en AtomicNode y CompositeNode, que en síntesis observan un entidad atómica a nivel de hipermedia, o toda una jerarquía de composición de objetos, respectivamente.

Nodo compuesto (CompositeNode)

Un nodo compuesto representa una entidad compleja, es decir una agregación de partes, capturando una organización jerárquica de la información, no basada en links sino en composición. Esta relación es extraída de la aplicación subyacente, mapeando un objeto compuesto, y todos los que representan sus partes. Los objetos "parte" ya deben estar mapeados en otros nodos, atómicos o compuestos, para ser luego agregados al nodo compuesto padre.

Nodo anotación (Annotation)

Un nodo anotación es un nodo con único dato de texto.

Un nodo anotación no aparece entonces en el conjunto de nodos de la hipermedia; el que aparece es su dueño, que mantiene una referencia hacia él por intermedio del ancla que lo activa. A pesar de esto se lo considera un nodo porque puede constituir el origen de un link. La diferencia con el resto de los tipos de nodos es que no puede constituir el destino de un link más que de aquel que nace en su nodo dueño.

Link (HyperLink)

Los links son los componentes esenciales de una hipermedia junto con los nodos, y al mismo nivel de importancia que ellos. Son los elementos que hacen factible la navegación, característica principal de una aplicación hipermedia.

Los links relacionan dos o más nodos, el primero llamado origen del link y el o los segundos llamados destinos o punta de link. Se considera que un link puede tener sólo un nodo origen, pero pueden existir varias anclas dentro del nodo para activar dicho link.

El link es el responsable de activar o abrir el nodo destino cuando él mismo es activado. En el momento de activarse, el link puede ejecutar ciertas acciones que tengan que ver con la navegación o con alguna poscondición al partir del nodo origen. Estas acciones pueden especificarse en un bloque o como un mensaje a ser enviado al nodo origen.

Para poder dar cierta semántica a los links, éstos tienen un atributo que define su significado. Este significado está redefinido en las subclases de HyperLink, como veremos posteriormente.

Volviendo al destino del link, existen distintas variantes para su activación. Además de ser simple o múltiple, el destino de un link puede ser fijo, cuando se especifica explícitamente el/los nodos a los que arriba, o computarse dinámicamente al momento de la activación, entre otros. Para evitar tener que testear por el tipo de destino, se separó la estrategia de obtenerlo en una jerarquía aparte de "resolvedores de destino" (EndpointSolver), con subclases específicas para cada forma distinta de activación. Por otro lado, la cardinalidad también requirió una jerarquía distinta de "destinos de link" (LinkEndpoint), subclasificada en "destino simple" (SingleEndpoint) y "destino múltiple" (MultipleEndpoint). En la Figura 8 vemos la jerarquía que comienza en HyperLink, y su relación con EndpointSolver y LinkEndpoint, las que serán explicadas más abajo.

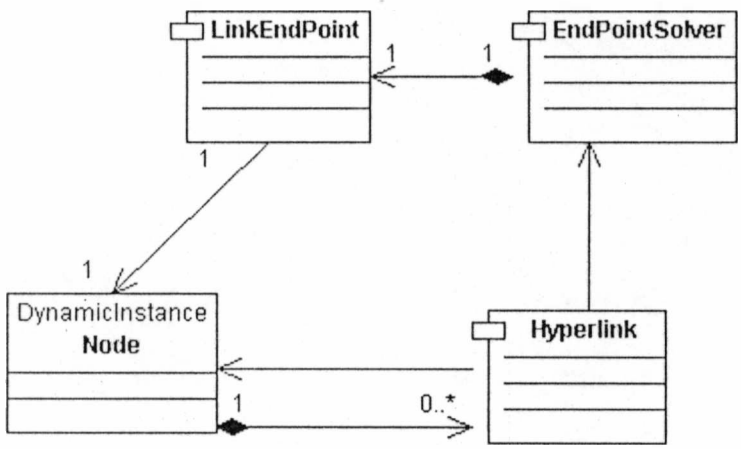


Figura 17. HyperLink y su relación con los colaboradores EnpointSolver y LinkEndpoint.

La clase HyperLink es una clase abstracta que factoriza las características arriba mencionadas para sus subclases. Al igual que ciertos nodos "observan" objetos de la aplicación, ciertos links también observan y mapean las relaciones entre esos objetos. Esto provoca la subclasificación de esta clase en links observadores o "clasificados" y links "no clasificados". Los links clasificados obtienen tal denominación del hecho de poseer una "clase de link" que define sus características, de la misma forma que la clase de nodo lo hacía para los Nodes.

En resumen, las responsabilidades de esta clase son:

mantener su origen;

mantener su destino y activarlo cuando el link es activado;

mantener su significado;

mantener las acciones involucradas en la navegación, o poscondiciones.

Indice (Index)

Esta estructura provee un acceso directo a sus componentes. Presenta un listado donde por cada entrada se muestra la concatenación de datos determinada por los selectores sobre un nodo target que cumplen el predicado de selección.

Clase de nodo (NodeClass)

Una instancia de NodeClass, como se dijo anteriormente, simula ser una clase que contiene la estructura y comportamiento de sus instancias, pero esta estructura y comportamiento serán las que se observan de la aplicación subyacente que se está extendiendo.

La incorporación de esta clase tuvo como propósito principal evitar la subclasificación de Node por cada clase de la aplicación que se quisiera observar. Esta subclasificación era necesaria principalmente por dos razones:

especificar los aspectos a ser observados sobre un grupo de objetos de la misma clase (y por lo tanto que comparten esos aspectos), para un cierto grupo de nodos;

permitir la automatización en la creación de nodos, y así evitar que por cada objeto de la aplicación se creara su nodo observador "a mano".

A pesar de tener estas necesidades, subclasificar Node era una muy mala solución, dada la terrible explosión de clases que esto provocaría por cada nueva aplicación (una sola aplicación hipermedia generalmente involucra muchas clases de nodo, y la cantidad se multiplica en cada aplicación diferente). Además, esta subclasificación estaría en manos del usuario del framework, que en cada distinta instanciación del mismo necesitaría crear nuevas clases. Esto implicaría que el usuario debe conocer detalladamente la implementación del framework para poder extenderlo, lo que lo convierte en un "framework caja blanca" [Johnson+88].

Otro problema es que la clase Node es subclasificada por otro criterio, como vimos anteriormente, lo que confundiría las distintas dimensiones de subclasificación.

Bajo estas circunstancias, la solución fue crear una clase distinta, que llamamos NodeClass, y hacer que sus instancias actuaran simulando clases para cierto grupo de nodos. Este diseño sigue el propuesto por el patrón "Type-Object" y su extensión, el Dynamic Template (ver Sección 5.3.1).

Las ventajas que se lograron por sobre la solución de subclasificar fueron:

El usuario sólo necesita especificar las clases de la aplicación a ser observadas, y los aspectos a mapear de cada una; por cada clase a observar se crea una instancia de NodeClass, y automáticamente todas las instancias de Node (en realidad de sus subclases), necesarias para mapear cada uno de los objetos de aquella clase. Tanto la nodeClass como sus Nodes mantendrán, a distinto nivel de especificación, los aspectos mapeados.

Por cada objeto de la aplicación que es creado dinámicamente durante la ejecución de la hipermedia, se crea automáticamente el nodo asociado si corresponde; esto es posible porque la clase de nodo observa la creación de cada nueva instancia de su clase asociada;

Una clase de nodo puede mapear más de una clase de la aplicación. De esta manera se permite flexibilizar el diseño de la hipermedia con respecto al diseño de la aplicación, que no había sido pensado para ser "navegado". La navegación implica una serie de consideraciones en cuanto a eficiencia, presentación de la información en forma inteligible, cuidado de no provocar desorientación, etc., que podrán ser tenidas en cuenta al crear las distintas clases de nodo.

El hecho de que los nodos que mapean objetos de la aplicación sean compuestos de distintas vistas requiere que una clase de nodo también se componga de la misma forma, dado que los nodos son "clones" o copias de ella. De esta forma aparecen las clases de vista de nodo (InContextViewClass) que actúan como clases de nodo pero para las vistas de los mismos. Una clase de nodo es entonces una agregación de clases de vistas de nodo.

Resumiendo lo dicho, las responsabilidades de esta clase son:

- conocer la/s clase/s de la aplicación que observa;

- conocer toda la información necesaria para crear los nodos que observen objetos de la clase de la aplicación relacionada (hipermedia a la que pertenece, aspectos a ser observados, clases de vistas de nodo, anclas de links);

- crear los nodos necesarios por cada objeto de la aplicación perteneciente a la clase relacionada;

- mantener los nodos que siguen su patrón, para permitir la modificación de ellos ante cambios o agregación de datos;

- mantener las clases de vistas de nodo, y delegarles a ellas la creación de las vistas para cada nuevo nodo que debe ser creado.

Clase de Link (LinkClass)

Idénticas razones por las que fue necesario introducir la clase NodeClass motivaron la definición de la clase LinkClass. Es decir, la misma relación existente entre NodeClass y Node se presenta entre LinkClass y ClassifiedLink. La diferencia existente es que LinkClass no observa una clase del nivel de objetos sino una relación entre clases, representada mediante el origen, destino, significado y poscondiciones, que respetarán todos los links pertenecientes a esta clase. El origen de una linkClass es siempre una nodeClass y de esta manera un link será creado a partir de cada nodo de la nodeClass origen.

Una clase de link que mapea una relación del nivel de objetos tendrá necesariamente como destino otra clase de nodos, y la especificación de cuáles nodos resultan relacionados estará dada mediante un predicado que involucra a los objetos representados en los nodos origen y destino. De esta manera, una linkClass creará un classifiedLink por cada nodo origen automáticamente al ser especificada, pero recién en el momento en que el link es activado se computa el destino según aquel predicado. Así como la especificación del destino de un link se mantiene en una instancia de LinkEndpoint en colaboración con un endpointSolver, una clase de link colabora con una 'clase de destino de link' (LinkEndpointClass) para mantener la especificación del destino para todos los links clasificados que le pertenezcan. En el caso en que el destino sea fijo, este se mantiene directamente en un LinkEndpoint. Por último, en el caso en que el destino deba crearse en el momento de la activación, lo que se mantiene es una instancia de EndpointFactory con la especificación de lo que se debe crear.

En el nivel de hipermedia puede surgir la necesidad de definir una relación entre una clase de nodos y un nodo no clasificado en particular. Por ejemplo de la clase de nodo 'Salas' de un museo, al hyperNode que muestra el mapa de ubicación de las salas. Este tipo de relación, a pesar de que no surge del nivel de objetos, también es representada por una linkClass que, como siempre, crea un classifiedLink desde cada nodo origen. El destino ya no será computado sino fijo en el nodo pertinente.

Resumiendo lo anterior, las responsabilidades resultantes de esta clase son:

- conocer la clase de nodos origen;

- crear y mantener el destino de los links;

- mantener el significado de la relación;

- mantener las poscondiciones de navegar los links que le pertenecen, y activarlas;

- crear y mantener los links que pertenecen a sí misma, cloneando sus datos.

Hipermedia (Hypermedia)

Cada instancia de esta clase conformará una aplicación hipermedia distinta. Esta clase agrega todos los nodos, links, clases de nodos, clases de links, historia, y además conoce el primer nodo que se abrirá al abrir la aplicación hipermedia.

Las responsabilidades de esta clase son:

- mantener su identificación, que por lo general será su nombre;

- proveer el protocolo necesario para crear clases de nodos, clases de links, nodos y links no clasificados;

- proveer el protocolo para crear contextos y transiciones entre ellos;

- abrirse iniciando una sesión activando el primer nodo;

- mantener la historia de nodos visitados.

Manejador de la historia de nodos visitados (HistoryManager)

La historia de nodos visitados provee, como lo implementado en distintos ambientes, una lista ordenada con los nodos por los que el lector ha ido navegando y un acceso directo a ellos. La definición de HistoryManager en el framework permite además configurar la historia para que registre sólo los nodos visitados, o sólo los links activados, o ambas cosas. Si el elemento a registrar es un nodo se guarda con él el contexto y la representación en la que fue accedido.

Las responsabilidades de esta clase son:

- registrar los componentes de hipermedia visitados en una sesión;

- mantener la configuración de los objetos navegacionales a registrar;

- reinicializarse;

- filtrar la historia con cierto predicado de selección.

Grafo de contextos

Un grafo de contextos mantiene, para una hipermedia, los distintos puntos de vista o perfiles de usuario que accederán a la misma y que requieren una visión especial de ella. Por cada contexto o perfil se conocen además los restantes contextos a lo que un usuario con ese perfil podría cambiarse además del que ha sido definido expresamente para él. Los posibles cambios de contexto se representan mediante instancias de ContextTransition. El grafo de contextos está formado por los mismos Contextos y las transiciones a otros contextos.

Transición de contexto (ContextTransition)

Cada instancia de esta clase representará un arco en el grafo de contextos. Esta clase provee además la posibilidad de definir acciones asociadas con el cambio de contexto, como podría ser reestablecer la historia de nodos visitados. Cuando un usuario decide cambiar de contexto la transición correspondiente es habilitada y esto produce la ejecución de las acciones asociadas.

Vista de Nodo en Contexto (inContextNodeView)

Una vista de nodo representa la forma en que un nodo de múltiples vistas se mostrará bajo cierto contexto o contextos. Una vista para un nodo particular se puede asociar a más de un contexto cuando la misma información será presentada a distintos perfiles de usuario. La siguiente figura muestra la relación existente entre el grafo de contextos y las vistas de nodos.

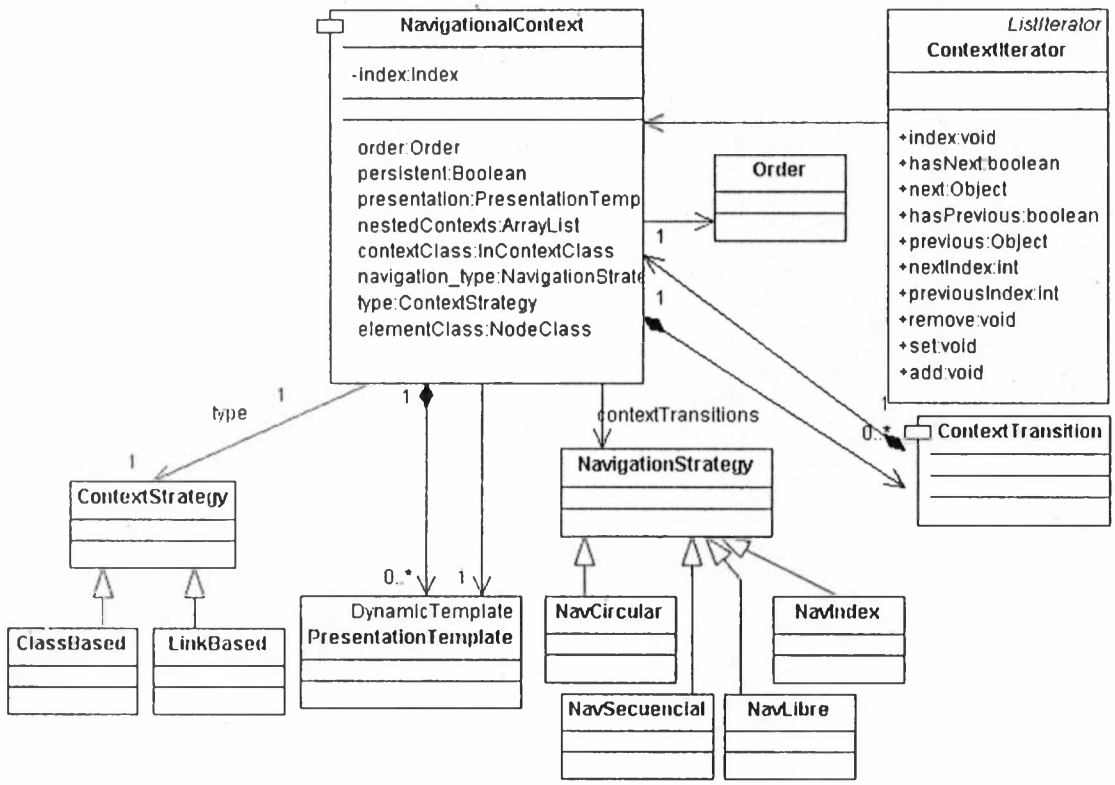


Figura 18. Contextos y Vistas de Nodo

Una instancia de InContextNodeView mantiene los datos a mostrar, acciones que se podrán activar y anclas de links que dependen de uno o varios contextos en particular, con los que está asociada. Esto permite que la información de la entidad mapeada en el nodo se filtre dependiendo del perfil del usuario.

En el proceso de navegación, el contexto actual se va pasando como parámetro entre los nodos y links que se activan. Cuando un nodo recibe el mensaje de activarse, busca entre sus vistas la que corresponda al contexto actual y la activa.

Una vista es además una agregación de 'representaciones'. Una representación describe cómo se mostrarán los datos que la vista estipula. Se podría decir entonces que la vista denota el qué se va a mostrar y la representación el cómo se va a mostrar. Describiremos el concepto de Representación en el siguiente ítem.

Hacer de "mediador" (con el significado del pattern Mediator en [Gamma+95]) entre la representación activa y el nodo al que pertenece, ante el pedido de datos o comportamiento por parte de la representación y que no se encuentran en la vista particular sino en el nodo, por ser información de todas las vistas.

Clase de Vista de Nodo (InContextViewClass)

Sabemos a este punto que las instancias de NodeClass representan clases para los Nodes y son las encargadas de crear instancias de estos últimos. Para esto una NodeClass guarda la especificación necesaria de las características que formarán a un nodo clasificado por ella. Entre las características que conforman un nodo están sus vistas, como vimos en el ítem anterior. La especificación de una vista de nodo a nivel de InContextNodeClass se guarda en una instancia de InContextViewClass. A esta instancia se le delega además la responsabilidad de crear una instancia de InContextNodeView que responda a su especificación. Resulta entonces que instancias de InContextViewClass pueden considerarse como tipos para las instancias de InContextNodeView, aplicando nuevamente el patrón Dynamic Template.

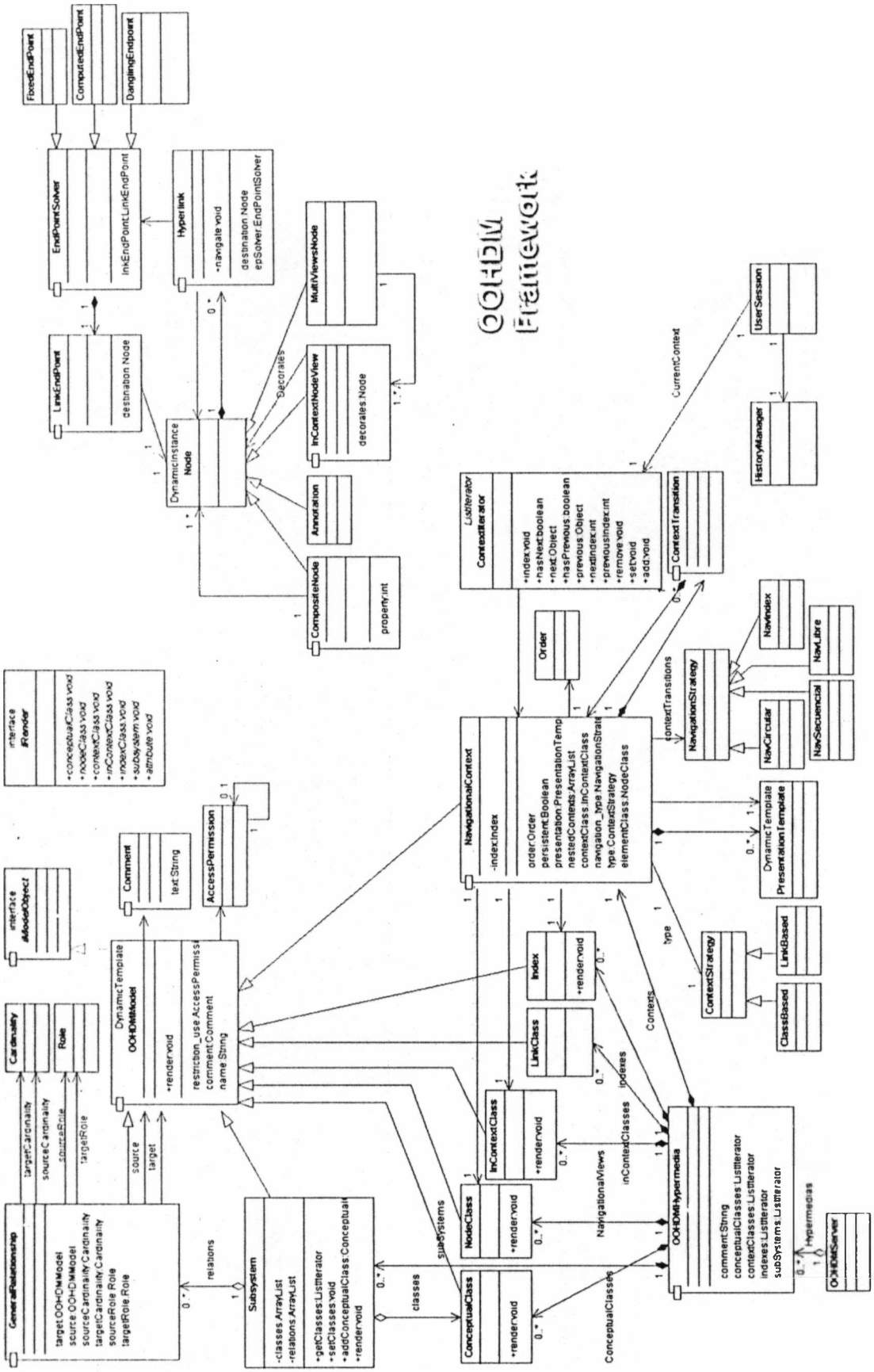


Figura 19. Diagrama general del framework OOHDM.

5.3 Patrones Utilizados en el Framework

5.3.1 El Dynamic Template

Una hipermedia está compuesta de nodos y relaciones (nodes and links). Bajo esta metodología, se modela el dominio (Diseño Conceptual) usando los famosos principios de modelación orientados a objetos. La siguiente fase define la estructura de navegación (toma en cuenta el perfil del usuario y la tarea y se enfoca en los aspectos cognitivos y arquitectónicos de un sistema hipermedia), estableciendo las clases Node y Link.

Finalmente, también se cuenta con el diseño de interface para cada clase de nodo, donde el diseñador fija la apariencia de cada clase de nodo. Una vez que se termina el diseño, todavía no se cuenta con una hipermedia, porque no tiene ni nodos ni links, sino descripciones de sus clases, su interface y contextos de navegación.

Con esa representación, se puede crear una instancia de cualquier clase de nodo y llenarla con información (por ejemplo, una instancia de un pintor: "Van Gogh", y después de esa muchas otras instancias de la clase "pinturas" para cada obra de arte de Leonardo).

Ahora se tiene una hipermedia que es "independiente de la plataforma" que puede ser mapeada a HTML, Asymetrix's Toolbook, Director, etc. Se debe poder modelar cada componente de diseño como una clase y soportar cualquier tipo de operaciones que se le hagan, como si fuera una clase real. Las operaciones a una clase pueden involucrar agregar atributos de clase e instancia, herencia (relaciones is-a) y composición (relaciones part-of).

Los problemas planteados en este contexto son múltiples, y entre ellos surge el siguiente interrogante: ¿cómo creamos clases de nodos sin provocar una explosión en la jerarquía de NodeClass?

Los factores que pesan en el contexto anterior son:

La definición de diferentes subclases de NodeClass por cada conjunto de nodos que mapean instancias de la misma clase del modelo de objetos terminaría en una explosión de clases, la que sólo será un espejo de aquel modelo.

Es preferible instanciar clases existentes en el framework a crear nuevas y múltiples clases cada vez que el framework es usado.

Cuando una nueva instancia de una clase de la aplicación mapeada al nivel de hipermedia es creada, un nuevo nodo debería ser creado para mapear esa instancia.

Todas las fuerzas mencionadas arriba en cuanto a los nodos se aplican de la misma forma a los links.

Cada nodo puede tener diferentes vistas o "caras" con las que se muestra dependiendo del contexto actual. Esto implica que al crear un nodo de cierta clase deberían crearse con él las vistas de nodo que su clase define.

Solución:

El patrón Dynamic Template [Lyardet97] fue utilizado de la siguiente manera:

Se definió la clase *NodeClass*, cuyas instancias cumplen el rol de clases para las instancias de las subclases de *Node*. Cada *NodeClass* se relacionará con una o más clases del modelo, de las que observará la creación de instancias para crear Nodes

asociados automáticamente. El conjunto de *NodeClasses* definidas delinearán el modelo de datos y comportamiento de la hipermedia.

Para manejar el hecho de que un *Node* es una agregación de vistas fue necesario aplicar el mismo pattern nuevamente en forma anidada. De esta forma, se definió la clase InContextClass cuyas instancias actúan como clases para las instancias de InContextView.

En este punto, pedimos prestado del Type Object, la estrategia de tener clases separadas cuyas instancias se comportan tanto como clases como también como instancias. Ya que es muy difícil (o imposible) agregar comportamiento a una clase en tiempo de ejecución, lo que creamos no son clases "reales" sino objetos 'template'. La estructura de estos 'templates' es definida dinámicamente, y pueden crear instancias de ellos.

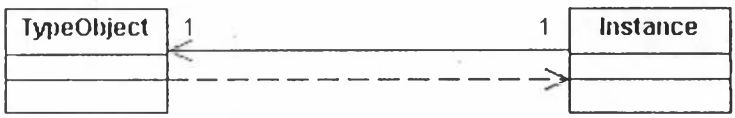


Figura 20. Diagrama de clases del Type Object.

El pattern "Dynamic Template" describe cómo las clases creadas dinámicamente pueden ser extendidas a través de la herencia o agregando atributos de clase e instancia, y la forma en que las instancias de estas clases son generadas. También describe cómo estas últimas se comportan como instancias reales cuyo tipo y número de atributos puede variar en tiempo de ejecución.

Para ofrecer un mecanismo dinámico para agregar atributos a una clase, necesitamos describirlos de forma que la creación de las instancias de clase con los atributos especificados pueda ser posible. Estas descripciones de los atributos podrían ser aún más especializadas, para especificar si los atributos serían atributos de clase o instancia en el momento de ejemplificarlos concretamente

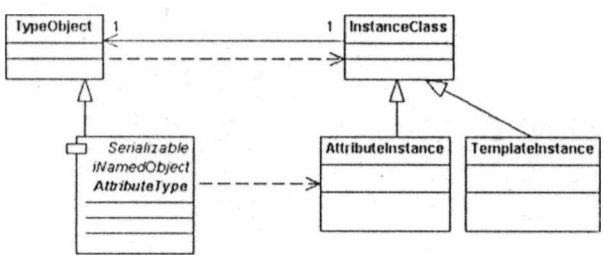


Figura 21. Diagrama de clases del Dynamic Template.

Las clases, cuyas instancias representan instancias de sistema real, son todas descendientes de la InstanceClass, como AttributeInstance y TemplateInstance. La primera es la contraparte AttributeType y mantiene el valor real de un valor de atributo de instancia, mientras que la segunda representa las instancias de una clase.

La clase AttributeType ofrece una forma de describir un atributo de clase y funciona como el método "Factory" dependiendo del valor TypeDescriptor. Cuando se crea una instancia de una clase dinámica, la clase le pide a sus atributos una instancia de sí mismos que pertenecerá a la nueva instancia.

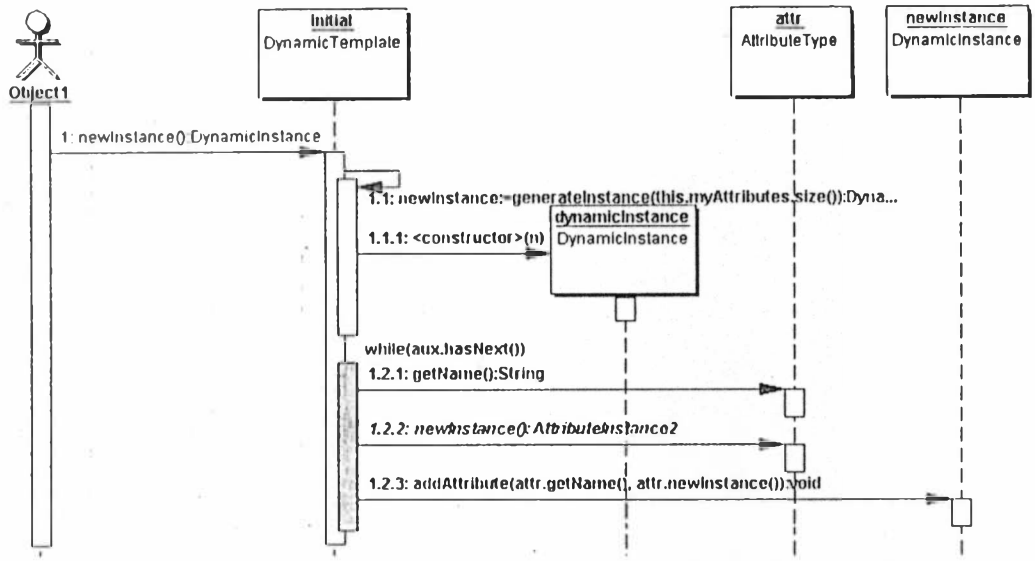


Figura 22. Diagrama de interacción mostrando la creación de instancias a partir del DynamicTemplate.

La idea de soportar la herencia única a través de un mecanismo de subclase dinámico se basa en la habilidad de crear una concatenación virtual de los atributos de una clase con los atributos de su superclase.

Para ofrecer esta característica, toda instancia DynamicTemplate tiene:

Sus atributos de instancia y clase guardados en diccionarios.

Un método que dice cuántos atributos tiene la clase (no importa si son heredados o no) (-attributesCount-)

Una referencia a su superclase.

Cuando se fija la superclase de una de las clases dinámicas, la nueva subclase fija su "super" referencia a la clase dada. En esta fase, podríamos establecer el pattern "Observer" (GHJV95) que notificaría sus instancias y subclases.

Todas las clases e instancias creadas dinámicamente de estas clases tienen acceso a sus atributos a través de un método para especificar si los atributos deben ser buscados por nombre o por índice.

Este enfoque permite esconder si los atributos son heredados o no ofreciendo un mecanismo de acceso uniforme. Tanto la clase DynamicTemplate como TemplateInstance implementan los métodos AttributeBy... para ganar acceso a sus atributos, con ciertas diferencias:

En la clase DynamicTemplate, estos métodos proveen acceso a los atributos de clase buscando a través de sus superclases si es necesario, y los métodos InstanceAttrBy... para acceder a las definiciones de atributos de instancia.

TemplateInstance provee acceso a sus atributos sin la necesidad de una búsqueda extra ya que como instancia, ya tiene todos sus atributos.

Participantes

Dynamic Template (TypeOf)

Es la clase de Dynamic Template.

Tiene una instancia separada por cada nueva clase dinámica creada.

Soporta la herencia única entre instancias.

Soporta el ganar acceso a los atributos por nombre o índice.

TemplateInstance (InstanceClass)

Es una instancia de Dynamic Template

Implementa el mecanismo para soportar el acceso a los atributos por su nombre o índice.

AttributeType (TypeOf)

Especifica el nombre y describe un atributo Dynamic Template;

Hace caso omiso del método createInstance y devuelve una instancia de AttributeInstance.

AttributeInstance (InstanceClass)

Sus instancias representan los atributos de una instancia de clase y mantienen los valores.

Consecuencias

Las ventajas del pattern Dynamic Template son:

Permite agregar templates dinámicamente como también agregar o borrar atributos de 'template' o de instancia, y establecer su superclase en tiempo de ejecución.

Permite al Objeto cambiar su Dynamic Template.

Permite a un Dynamic Template cambiar su superclase.

Da al sistema una capacidad muy flexible de manejar y aumentar el manejo de información.

Ya que el pattern Dynamic Template ofrece la misma separación entre objetos que se comportan como clases y objetos que deben ser instancias de clases, el pattern Type Object está "incluido" dentro del pattern Dynamic Template. Por lo tanto, sería posible usar este pattern como un Type Object en contextos similares para ofrecer funcionalidad extra. Sin embargo, el uso principal de este pattern es en el contexto de sistemas altamente dinámicos, como aquellos presentados en la sección motivación.

Las desventajas del Dynamic Template Pattern son:

El Dynamic Template Pattern comparte las mismas desventajas del pattern Type Object más cierta incomodidad para manejar atributos a través de un método en lugar de sólo mandar un mensaje. Esta limitación se debe a la necesidad de hacer cierto cómputo para buscar el atributo deseado.

El mecanismo de herencia funciona sólo con los objetos Dynamic Template. Es decir, no se puede fijar una clase arbitraria para que se convierta en su superclase. Debe ser un descendiente Dynamic Template.

El acceso al atributo es más lento que el acceso variable normal.

Aunque parte de la estructura de un objeto Dynamic Template puede ser fijado estáticamente, los atributos especificados así no pueden ser heredados. Por lo tanto, para explotar el rasgo de herencia y acceder a los atributos de forma uniforme (sin tener que distinguir entre atributos definidos estática y dinámicamente), todos los atributos deberían ser definidos dinámicamente.

En base a la desventaja anterior, los objetos Dynamic Template y TemplateInstance no pueden determinar estáticamente sus estructuras.

Es difícil determinar si se está usando una variable en un Dynamic Template ya que no podemos simplemente buscar el código de fuente.

Implementación

Existe un número de cuestiones que considerar cuando se implementa el *DynamicTemplate* pattern:

La instancia de clase hace referencia al *DynamicTemplate*: ya que el lenguaje de programación no conoce la relación entre un objeto *TemplateInstance* y un objeto *DynamicTemplate*, la relación y significado de la referencia de clase de classInstances debe ser establecida por el diseñador de software.

Cambiar dinámicamente la clase de instancias: cambiar la clase de una instancia puede significar un esfuerzo extra, ya que la idea de crear una clase completa no nos permite asumir que la nueva clase tendrá una estructura similar (como sucede con el type object). Tenemos que asegurarnos que los atributos de instancia sean consistentes en significado y estructura con la clase. La mutación de los atributos podría ser una posibilidad, ajustando los atributos de instancia con los nuevos atributos de clase por nombre, y agregando/borrando los atributos restantes.

Reflejar los cambios de Clase a sus instancias: el reflejar los cambios de clase a sus instancias es una responsabilidad de clase. Lenguajes como Delphi o C++ no proveen a sus clases de Objeto básicas la habilidad para mandar un mensaje "auto cambiado". Por lo tanto, debería implementarse esa capacidad ya sea a través del DynamicTemplate mismo o delegando esta función en objetos especializados que se comportan como "Observers" o DependencyTransformers [Woolf94] (ambos deberían ser programados también).

Habilidad de los atributos para mantener valores: las instancias AttributeInstance tienen la habilidad de mantener la información del atributo real (valor). El problema surge con los lenguajes OO estáticamente tipeados como C++, donde la información tipo es necesaria en tiempo de compilación. Una forma fácil de superar este problema sería el siguiente:

Agregar un símbolo (identificador) al AttributeType que especifica el tipo del atributo.

Agregar los métodos asString, Asinteger, etc. a la Clase AttributeInstance y derivar nuevas subclases para soportar el valor para todos los tipos especificados.

El método createInstance del Attributetype funcionaría como un método 'factory', para ajustar la creación de la instancia de atributo con el tipo de atributo.

5.4 Soporte para mantener el estado de las sesiones de los usuarios

Cuando los usuarios navegan en el web utilizando un browser, los detalles del Protocolo de Transporte de Hipertexto (http) que hace posible esa experiencia transcurre cómodamente inadvertida. http es una especificación no orientada a conexión (connectionless) que establece como se realiza la comunicación entre el browser y un servidor de web. Los protocolos "no orientados a conexión" poseen ciertas ventajas no obstante, presentan un problema al momento de mantener

información o "estado" acerca de los usuarios que se encuentra visitando el website o hipermedia.

Desde la perspectiva de un servidor de web, cada pedido a través de http es independiente y distinto de los demás pedidos realizados. Eso significa que la información que un usuario ingresa en una página (a través de un formulario por ejemplo) no se encuentre automáticamente disponible en la próxima página que se acceda. a menos que el servidor de web lleva acabo las acciones necesarias para que ello ocurra. }el desafío es entonces como identificar cuales pedidos, de los miles que recibe un servidor de web, provienen del mismo usuario. Una forma es a través de "cookies".

Implementación del soporte de estado a través de "cookies".

Una cookie es una pieza de información que el cliente almacena por parte del server. Es este caso, la información almacenada en la cookie es originada en el server, y es retornada al cliente como parte de la respuesta del server a un pedido http. Un cliente puede contener varias cookies almacenadas simultáneamente, y cada una de ellas puede estar asociada con un website (o página)particular. El servidor de web puede entonces utilizar la información en la cookie para identificar al usuario y, dependiendo de la naturaleza de la información recolectada, la posible personalización de la apariencia de una página web. El servidor puede también agregar o cambiar información antes de retornarla al usuario.

Las cookies tienen una fecha de expiración. Si la fecha de expiración de una cookie ha sido explícitamente establecida para algún momento en el futuro, el browser automáticamente salva la cookie en el disco del cliente. Las cookies que no tienen una fecha de expiración son válidas solamente durante la vida de una sesión del browser, de modo que en cuanto el browser es cerrado, la cookie es borrada de la memoria de la computadora.

Dado que la cookie es retornada server con cada pedido http, las cookies son una buena manera de identificar una serie de pedidos que provienen del mismo usuario. Cuando un pedido de un usuario conocido es recibido, el identificador único puede ser extraído de la cookie y utilizado para recuperar información adicional sobre el usuario, que es en general mas sensible y privada, de una base de datos. De esa manera, la información privada es almacenada en forma segura en el server y no en la cookie donde se encuentra expuesta al hacking. La cookie solo contiene la clave para la información privada almaceda en el server. Cuando un request es recibido sin la correspondiente cookie, o cuando la cookie no contiene el identificador necesario, se asume que el pedido proviene de un mismo usuario. En ese caso, un nuevo identificador es generado y retornado al cliente, y su información agregada a la base de datos del website.

Un servidor de web típico mantiene información acerca de los usuarios que visitan si website en su base de datos del usuario. Esta información de personalización de la información que suele mantenerse relativamente constante en el tiempo, tal como el nombre del usuario o la dirección de mail, o puede contener información cuyo tiempo de vida sea mucho menor, como el número de ítems comprados durante la sesión actual.

En la terminología de Internet, la palabra "sesión" se refiere al tiempo durante el cual un usuario especifico se encuentra accediendo activamente a los contenidos de un website. Una sesión comienza cuando el usuario accede a la primera página de un

sitio, y finaliza cuando lo deja la ultima página del mismo. La piezas de información específica del usuario, relevantes a una sesión particular son generalmente conocidas como "estado de la sesión". Algún tipo de técnica de cookie es a menudo utilizada para identificar los cuáles pedidos proviene de la misma sesión de usuario (cabe notar que cada browser que el usuario abra en su desktop establece una nueva sesión que identifica unívocamente no sólo al usuario sino también a la instancia del browser).

Manejo de la sesión en forma centralizada.

Las desventajas de mantener el estado de la sesión en el cliente son la seguridad y la limitación de tamaño que presenta. Dada que la información se encuentra almacena en un lugar conocido, se halla expuesta al husmeo (sniffing) o robo pr parte de otra aplicación. Por otro lado, el problema de la limitación del tamaño de la cookie es una restricción importante. EL máximo tamaño permitido en una cookie varía según el browser, si bien todos los browsers permiten al menos 4Kb. Finalmente, debe tenerse en cuenta que la posibilidad de utilizar los cookies siempre se encuentra limitada por la potencial decisión del usuario configurar su browser para no permitir el uso de cookies por motivos de seguridad, lo cual impediría su eventual acceso y utilización de aquellos websites que descansen en este mecanismo para si funcionamiento y customización de la información.

Existen numerosos casos en los cuales la utilización de las cookies es suficiente, sin embargo, en la medida que los websites y su funcionalidad comienzan a poseer una funcionalidad mas sofisticada, necesitan almacenar información que a menudo no puede realizarse en las cookies. Para esos casos, los diversos servidores de web y generadores de págnas dinámicas tales como ASP(Active Server Pages), JSSP(Javascript Server Pages) o JSP (Java Server Pages) proveen de mecanismos que automáticamente proveen la posibilidad de mantener la información de estado en forma centralizada, y que es accesible por el desarrollador a nivel en los scripts que codifican la generación de las páginas.

La administración centralizada de las cookies posee numerosas ventajes, entre las que podemos nombrar a la reducción del tráfico de red, soporte de cualquier tipo de usuario sin importar si soporta cookies o no, ni las limitaciones de las mismas.

Administración del estado de una sesión a través de múltiples servidores.

Habitualmente se denomina "webfarm" aquellos websites que cuentan con un conjunto de servidores para su funcionamiento. La necesidad de utilizar múltiples servidores aparece cuando el volumen de usuarios que acceden simultáneamente a un determinado website alcanza un gran número. Los pedidos del http son usualmente ruteados para cada uno de los servidores con una estrategia round-robin, para distribuir la carga y permitir que el website reciba mayor cantidad de pedidos.

Mientras que una el soporte de manejo de una sesión en el server funciona bien en todas las tecnologías de generación de páginas dinámicas, este soporte no es muy útil en una webfarm dado que solamente permite mantener el tracking de la información en un servidor de Web que es el que maneja el pedido en particular. A medida que numerosos pedidos son enviados a diferentes servidores de web, la información almacenada e el primer pedido no se encuentra disponible dado que reside en otro server.

Una estrategia de implementación de cookies centralizadas en una webfarm es manteniendo el state en una base de datos relacional.

A medida que se agregan servidores a una webfarm, se genera mayor congestión en el acceso a la base de datos que contiene la información del estado de las sesiones generando además, un mayor tráfico en la red. De este modo, el mayor número de pedidos resueltos por segundo queda acotado por el número de pedidos resueltos por la base de datos.

Las diferentes tecnologías de servidores de transacciones de componentes hacen hincapié en la necesidad de utilizar componentes "sin estado" o "stateless" para lograr mayor escalabilidad. Esta característica se debe a que dicha tecnología se encuentra fuertemente basada en la tecnología existe de servidores de transacciones convencionales, limitando su funcionamiento en caso de utilizar componentes "con estado" o "stateful" a la mera serialización de los accesos al mismo con la consiguiente perdida de perfomance (en vez de permitir políticas agresivas como "Many Readers, single Writer", solamente implementan exclusión mutua fair).

Cuando el volumen de la información de las cookies es reducido, habitualmente se utiliza una técnica drásticamente diferente para lograr mejor escalabilidad de los servicios provistos por el website: mantener la información de la sesión en la red (también conocido como "OnTheWire"). En este approach, la información que habitualmente se almacenaría en la cookie, queda embebida en campos ocultos de la página HTML que es enviada al usuario, la cual viaja desde el server al cliente y viceversa repetidas veces hasta que se completa una determinad tarea o hasta que el usuario sale del website. Si bien este approach es engorroso y genera un consume extra del ancho de banda disponible por el mayor volumen de información a transmitir, permite que los pedidos de cualquier usuario que accede al website sea atendido por cualquier servidor, sin necesidad de un manejo centralizado de la sesión y permitiendo que la interpretación de la misma sea realizada por componentes stateless que fácilmente escalan en el contexto de la tecnología que se encuentra actualmente disponible.


BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

6 Modelado de Procesos en Aplicaciones Web

6.1 Introducción.

La popularidad del Web brinda un canal inmenso de información y ha generado la aparición de una generación de aplicaciones que aprovechan la infraestructura Web, tales como las intranets y extranets como evolución de las aplicaciones cliente-servidor, y otras aplicaciones cuya existencia se debe a la Web.

Sin embargo, la naturaleza del Web basada solamente en documentos y links, no es suficiente para acomodar los requerimientos de aplicaciones funcionando en el mismo espacio que los documentos HTML. Un ejemplo de esto se evidencia en los links, cuya definición básica en el Web lo transforma en un vehículo polimórfico para cualquier cosa que no sea contener información (que es responsabilidad del documento). Es así que nos encontramos que los mismos links especifican navegación, descarga de información local, envío de información, operaciones sobre la información y operaciones de interfaz (aumentación por ejemplo, ver Typed Link).

Como consecuencia de este solapamiento conceptual, aparecen fenómenos como el problema de las herramientas navegacionales provistas por los browsers y su interacción con la aplicación Web que está mostrando. Un ejemplo es el problema de la acción del botón de regreso o "Back" en el medio de la realización de una tarea, que puede significar "deshacer el último paso" desde el punto de vista del usuario, descripta en [RossiHans01]. Frecuentemente las aplicaciones en el Web solucionan este problema impidiendo que los usuarios naveguen a dichas páginas, tal como se describe en el patrón Punto de No Retorno ("Point of No Return").

Sin embargo, soluciones como las descriptas no hace otra cosa que paliar una situación de endeble soporte del paradigma para la expresión de las aplicaciones y sus procesos. Dicha debilidad tiene su origen en el intento de modelar los procesos de negocios ("Business Processes") únicamente mediante las primitivas básicas del hipertexto: documentos y links, lo cual complica también el desarrollo de aplicaciones (ver modelo1).

6.2 Características de Navegación y Proceso de Negocios.

A continuación se detallan una serie de características que definen a la navegación y los procesos de negocios, para ayudar a comprender la naturaleza del problema:

Proceso

-Define un conjunto de actividades a ser ejecutadas y el flujo de control entre ellas.

-Tiene un estado que consiste en la actividad actual, incluyendo el conocimiento de si se encuentra activa o suspendida, y las actividades realizadas anteriormente.

-Como consecuencia, un estado implica las tareas que han sido realizadas, y también cuales actividades se deberán realizar subsecuentemente.

-La próxima tarea seleccionada surge como consecuencia de la evaluación del resultado de la tarea actual.

-Al entrar a una tarea, su apariencia depende del estado en el que se encuentra (inactiva, suspendida, etc.).

-La habilitación de los links "anterior" y "siguiente" dependerá de la tarea actual.

-El backtracking a una tarea anterior (en caso de estar permitida), no implica "deshacer" la última operación, sino volver a la actividad anterior en el estado en el cual se encuentra. La posibilidad de "deshacer" debe ser mostrada explícitamente.

-Para abandonar un nodo (salvo en el caso de "retornar" cuando esté disponible) estará sujeto a la terminación de la tarea.

-El link "siguiente" habitualmente está deshabilitado mientras no se cumplan las condiciones mínimas para poder continuar con la próxima tarea.

-Los links son habilitados condicionalmente.

-Una tarea puede disparar una secuencia de actividades que pueden o no contar con intervención del usuario.

Navegación

-Si separamos la noción de nodo de la navegación, entonces esta última es sin estado (o "stateless") mientras que el nodo puede estar "visitado" o "no visitado". El estado puede ser visitado o no visitado, y este puede o no mantenerse a través de las sesiones del usuario. esta dado por el nodo actualmente visitado en la pantalla.

-Desde el nodo actual no es posible inferir los nodos que han sido visitados anteriormente.

-Desde el nodo actual, no es posible establecer cuál será el próximo nodo a visitar. El usuario es quien decide el próximo link a visitar.

-El entrar en un nodo, lo que se ve dependerá del contexto que se está navegando.

-Los destinos de los links "anterior" y "siguiente" dependerán del Contexto actual en el cual el usuario está navegando. (Recordemos que en OOHDM un usuario siempre esta navegando en algún contexto).

-Para salir de un nodo no debe cumplirse ninguna condición, salvo la de seleccionar el link deseado.

-El backtraking a una página solamente significa volver a abrir el nodo en el contexto correspondiente y siempre se encuentra habilitado.

-El botón de "próximo" estará habilitado si el nodo actual no es el último en el contexto actual o si el contexto actual establece navegación circular.

-Salvo lo especificado en el punto anterior, no existe otra condición a cumplir para recorrer el link.

-Los links siempre están activos. Su estado refleja si el nodo destino ya ha sido visitado (en la sesión actual) o no.

-La navegación siempre requiere intervención del usuario.

6.3 Diseño Conceptual y navegacional de Procesos y Actividades en OOHDM

A continuación, continuando con la visión del desarrollo de aplicaciones hipermediales de OOHDM presentado en el capítulo 2, introducimos en 2 secciones separadas la definición de proceso de negocios y su contraparte en el esquema navegacional. Esta extensión a la metodología OOHDM fue presentada recientemente [RossiHans01].

Diseño conceptual: Modelado de Procesos y Actividades.

Para introducir los procesos de negocios, particionamos el diseño conceptual en 2 tipos de objetos: entidades y procesos. Los objetos entidades son relativamente permanentes en el dominio de la aplicación, , mientras que los procesos modelan los procesos de negocios que tienen lugar en ese dominio.

Podemos decidir acerca de si un objeto es una entidad o un proceso basado en las siguientes características:

Los objetos Entidades tienen un estado permanente y compartirlos en diferentes (instancias) procesos de negocios es complejo.

Los objetos de procesos tienen un estado temporal, Sus instancias pueden ser ejecutadas en paralelo sin inconvenientes.

Se comunican con los objetos entidades mediante el envío de mensajes.

Una actividad compleja, también llamada actividad padre, define una secuencia en la cual sus actividades deben ser ejecutadas para completar exitosamente un proceso.

Cada Actividad devuelve una página de salida antes de que lea el ingreso de datos que el usuario hizo en su página.

Considere por ejemplo las actividades del check out. El proceso "*Login*" crea una página en la cual el usuario incorpora o modifica sus datos personales, que "*Login*" procesa posteriormente. Esto es similar para el resto de las actividades.

Puesto que las entradas de información de las diversas actividades de un proceso de negocio como el check out están relacionadas y son utilizadas (e.g. para crear una orden), éstas deben ser consistentes. El proceso de negocio se ocupa de que toda la entrada de información se realice consistentemente, o el proceso no podrá ser completado por el usuario.

Hay otras maneras de lograr consistencia de entradas de información relacionadas. Una alternativa muy simple que se encuentra por ejemplo en www.amazon.com, que proporciona una gran página de entrada de información que incluye todas las entradas de información de las diversas actividades. Esto es una solución muy antipática para el usuario, y funciona solamente en el caso más sencillo en que un proceso no proporciona en su salida los datos la entrada de la cual depende la actividad siguiente.

Considere por ejemplo un ofrecimiento online de una póliza de seguro para su auto. En la primera actividad, se incorpora los datos relacionados con el seguro. La actividad subsiguiente le mostrará en su página de salida qué clases de seguros existen para su coche, y cuánto costará, de modo que usted pueda seleccionar el mas apropiado, etcétera.

El proceso dinámico de un proceso con actividades se describe en el siguiente. Cuando un proceso como el check out se comienza, la actividad del check out consigue el control. Puede ser que haga un cierto trabajo sí mismo y realice una interacción con el usuario. Pero en este caso, la actividad de "*CheckOut*" delega el trabajo en otras tareas mas pequeñas para ser realizada, crea como padre inmediatamente la actividad de "*LoginCliente*" como hija y la comienza. Ahora esto tiene control y hace su trabajo (que obra recíprocamente con el usuario). Una actividad hija puede acabar su proceso por varias razones como:

Puede terminar puesto que se termina su trabajo.

Puede suspenderse debido a un usuario entrado como la navegación fuera del proceso.

Puede interrumpir sí mismo puesto que el usuario cancela el proceso

Cuando una actividad del hija notifica a su padre sobre ella proceso que acaba y la razón de ella, el padre reasume control y continúa procesando; en este ejemplo que crea la actividad de "*ConfirmarItems*" iniciándolo, etcétera.

Como vemos, cada actividad tiene un estado. El estado de una actividad básica indica la punta actual del control y de la situación actual del control (puede ser activo o suspendido). En un sentido más general, el estado incluye las entidades que la actividad utiliza. Una actividad compuesta contiene una lista de las actividades hijas; su estado contiene además por lo menos la información si una actividad del hija esta actualmente activa, en la forma de un atributo "**actividadActual**" que apunta a la actividad hija si una tiene actualmente el control.

Esta información de estado es suficiente para un proceso de "*CheckOut*" con un flujo unidireccional de control secuencial. Sin embargo, "*CheckOut*" con un flujo de control que brinda al usuario una opción, requiere información adicional de cuales actividades que "*DirecciónEnvío*", "*OpcionesDeEnvío*" y "*OpcionesDePago*" se han ejecutado.

Puesto que una actividad hija devuelve el control a su actividad padre cuando termina su proceso (y no llama a actividad hermana), una actividad padre establece el flujo de control entre sus actividades hijas. De esta manera, hemos desacoplado las actividades de flujo del control entre ellas. Esto permite que las diferentes clases de actividades sean reutilizadas en otros diversos procesos. Por ejemplo la clase de actividad de "*OpcionesDePago*" puede ser utilizada al comprar CDs o pasajes aéreos, aunque el checkout de las compras del padre se diferencia totalmente del proceso que compra pasajes aéreos. Con respecto a la reutilización, ésta es una de las ventajas más atractivas de modelar explícitamente los procesos y sus actividades.

Diseño Navegacional: Mepeando Procesos a Nodos Actividad

Hemos discutido anteriormente que no podemos modelar procesos y actividades mediante nodos navegacionales. Por lo tanto, introducimos los nodos actividad ("activity nodes") en el esquema navegacional, y particionamos el espacio de diseño navegacional, en modo similar a lo realizado con espacio conceptual dividido en nodos entidades y procesos.

Mapeamos entidades del esquema conceptual en nodos entidades y los procesos en nodos actividad. Los nodos entidades son nodos hipermediales tales como han sido descrrptos en (...)

Un nodo actividad presenta la salida de una actividad y procesa su entrada. contiene botones como el "ok", "commit" o "cancel" (de dicha actividad), o ir a la siguiente (actividad) que lanzan acciones relacionadas al procesamiento de la entrada de la actividad y controla el progreso del flujo de control a la actividad siguiente.

6.4 Arquitectura del framework de workflow para OOHDM

Un buen framework provee funcionalidad madura dentro de un dominio específico. Dado que los frameworks requieren relativamente poco esfuerzo para cumplir las expectativas de sus usuarios, su diseño e implementación deben reflejar un profundo conocimiento del dominio.

Los frameworks bien diseñados capturan los procesos duraderos o *workflows* que son fundamentales para el dominio de aplicación que trata. Estos procesos duraderos capturan *workflows* que no cambian demasiado a lo largo del tiempo. Estos se ocupan de la secuencia de alto nivel de lo que se realiza (el diagrama de estados de "alto nivel"). Un beneficio del paradigma de workflow es que el mismo paradigma provee los medios para instanciar aquellos procesos dinámicos y detallados que se ocupan de la secuencia paso a paso de actividades necesarias para completar una tarea. La administración de workflows simplifica las relaciones complejas entre objetos que se encuentran en las aplicaciones orientadas a objetos de gran escala.

Los diseños y arquitecturas de software modernas recomiendan favorecer el concepto de composición de los objetos existentes en los sistemas versus la herencia [GOF] en el armado de toda la aplicación para maximizar la reutilización de la funcionalidad previamente construida. Se genera entonces una tensión entre la necesidad de modelar módulos de software con responsabilidades bien definidas y diseño orientado a la composición y al protocolo para maximizar la reusabilidad, y la necesidad de establecer las interacciones entre los mismos para generar la aplicación. Es entonces en estas complejas interacciones donde los mecanismos de workflow también minimizan la necesidad de código a medida y enfatizan la reusabilidad.

Arquitectura

Existe en general un consenso en lo que a la arquitectura del workflow se refiere. La primera aproximación se denomina Push Workflow Architecture, y consiste en tener un único "router" que envía cada Activity al paso correspondiente, en el momento adecuado dentro del ciclo de vida del Activity.

La segunda aproximación se denomina Pull Workflow Architecture, y en oposición a la anterior, utiliza un único repositorio o "pool" donde almacena cada Activity, hasta que es requerido por un paso determinado. Los pasos se subscriben al "pool" expresando su interés en ser notificados de la ocurrencia de ciertos eventos sobre Activity. Cuando estos eventos son señalizados, el "pool" le asigna el Activity en cuestión al paso que estaba interesado en la aparición de dicho evento.

Relación entre Activity y Workflow

Durante la permanencia de una Activity dentro del workflow, éste atraviesa diversos pasos. En cada paso, el Activity es sometido a diversos procesos, los cuales están íntimamente relacionados con la actividad que se realiza en dicho paso. En otras palabras, un paso puede verse como un conjunto de actividades, que se realizan tomando como partícipe primario al Activity que se encuentra activo en ese paso. Este conjunto de actividades que se ejecutan sobre una Activity, tiene generalmente

asociadas un conjunto de reglas, cuya verificación indica que la actividad ha culminado con éxito. Estas reglas no son otra cosa que condiciones que deben cumplirse en algún momento, durante la permanencia del Activity en el paso, y que cuando no son satisfechas, desencadenan un conjunto de operaciones tendientes a corregir las circunstancias que impidieron la validación de la condición asociada a la regla, de manera tal de que la regla finalmente sea validada.

Cuando una Activity llega a un paso, éste le proporciona los manejadores de excepciones para manipular las condiciones excepcionales que las reglas de dicho paso pueden encontrar. Esto puede verse como un proceso de suscripción del Activity al paso. Los Activities pueden pertenecer a diferentes "tipos". Un tipo de Activity responde siempre a una clasificación surgida del dominio específico de la aplicación subyacente al workflow, pero de todas formas su representación en esta modelización, se debe a la posibilidad de permitir el manejo de la misma excepción en forma diferente, dentro de un mismo paso, según el tipo de Activity sobre el cual se evalúe la regla. Cuando el Activity se subscribe al paso, el paso le asigna los manejadores de excepciones según el tipo del Activity en cuestión. Para esto el paso debe estar correctamente configurado por parte del usuario, de manera de tener discriminados los manejadores según los diferentes tipos de Activitys que puedan arribar al paso.

Una vez que hemos visto cual es la definición de una regla y como interactúan con los Activitys dentro del paso, veamos como encontrar una modelización en objetos, que nos permita representarlas.

Regla: una regla tiene la responsabilidad de evaluarse sobre una Activity verificando el cumplimiento de la condición asociada a ella. Cuando esta condición no se satisface, se "levanta" la excepción correspondiente. Dicha excepción es manejada desde el Activity utilizando el manejador que corresponda.

Evaluación de reglas

Otro punto interesante a tener en cuenta dentro de la modelización de un workflow, y más específicamente dentro de la representación de las reglas de negocio, es como se realiza la evaluación de las reglas sobre el Activity. Si bien el momento en el cual se comienzan a evaluar las reglas lo determina el paso, lo que el paso no puede determinar, es el orden de ejecución de dichas reglas, ni mucho menos, la forma en la cual se evaluarán las reglas sobre el Activity.

Con respecto al orden de ejecución de las reglas, éstas pueden evaluarse en forma secuencial, para lo cual alguien debe conocer el orden lineal en el cual estas reglas se evaluarán. En otras palabras, alguien debe poder determinar cual es la primera regla, la segunda y así hasta la última. Es más, podría suceder que este orden de ejecución se pueda ver alterado, como producto de la evaluación de una regla anterior. Es decir, puede pasar que una regla levante una excepción cuyo manejador, determine que el orden de ejecución normal de reglas, se altere de alguna forma en particular.

Con respecto a la forma en la cual se pueden evaluar las reglas, éstas pueden evaluarse en forma secuencial, es decir, una regla a la vez y en algún orden establecido, o por ejemplo, en forma paralela, es decir, varias reglas pueden tener que evaluarse al mismo tiempo sobre un mismo Activity. Aún más, puede que la forma de evaluación de las reglas sea una combinación entre estas dos formas anteriores.

Ahora bien, ¿cómo representamos este tipo de funcionalidad en nuestro modelo de objetos? En este caso, elegimos hacer uso de uno de los patterns de diseño

presentados en [ref a Gamma], más específicamente utilizamos un Strategy para representar las diferentes formas de evaluación de las reglas. En nuestra modelización introdujimos entonces la clase EvaluationStrategy, que es un a clase abstracta bajo la cual se encuentran representados los diferentes algoritmos de evaluación descriptos anteriormente.

Los pasos son configurados con un EvaluationStrategy en particular, con el cual colaboran para establecer la forma de evaluar las reglas. En realidad, el paso le delega a su EvaluationStrategy la tarea de evaluar las reglas sobre una Activity determinado. Por ejemplo, es el EvaluationStrategy quien determina cual es la primer regla a ser evaluada y cuando culmina la evaluación de dichas reglas.

Manejo de excepciones

Cuando una regla no es validada satisfactoriamente, se levanta una excepción. Dicha excepción debe ser manejada por el Activity para corregir las circunstancias que hicieron que dicha regla no se cumpla. Para esto, modelamos la secuencia de acciones a llevarse a cabo cuando una excepción se levanta, como un objeto más, instancia de la clase ExceptionHandler.

Por lo tanto, cuando una excepción se levanta, el producto determina cual es el manejador que corresponde ejecutar, y lo utiliza para que la regla sea finalmente satisfecha. Recordemos que cuando una Activity llega a un paso, este le asigna los manejadores de las excepciones que sus reglas pueden levantar, de acuerdo con el tipo de Activity de que se trate.

En la implementación de nuestro modelo, contamos además con una herramienta que facilita la instanciación de ExceptionHandlers, permitiendo la configuración de pasos con reglas y manejadores en forma dinámica. Esta herramienta es básicamente un editor de manejadores de reglas, con la cual el diseñador del workflow puede tanto construir nuevos manejadores como editar y modificar los manejadores ya existentes en el workflow.

Viaje de la transacción dentro del Workflow

Un workflow puede verse como un sistema con una entrada y una salida bien definidas, donde una transacción, en nuestro caso de tipo comercial, entra por el punto de entrada, atraviesa diferentes pasos, y sale por el punto de salida, habiendo experimentado cambios que son consecuencia de su viaje por dentro del workflow.

Para poder efectuar este viaje, es necesario que la transacción tenga la capacidad de realizar diferentes tareas, que le posibiliten precisamente atravesar cada uno de los pasos del workflow sin problemas. Para esto, nueva funcionalidad debe ser adicionada a la transacción, de forma tal de prepararla para su viaje por el workflow.

Ahora bien, queda claro que esta nueva funcionalidad solo es requerida durante la permanencia de la transacción dentro del workflow, y por lo tanto sería inadecuado adicionar protocolo nuevo a la transacción para satisfacer estos requerimientos de funcionalidad, ya que entonces estaríamos haciendo que este conjunto de nuevos mensajes, estén disponibles durante toda la vida útil de la transacción. Nosotros queremos que la transacción incorpore comportamiento nuevo solo mientras lo necesite.

Para esto utilizamos un Decorator [Gamma] sobre la transacción, que denominamos Activity. Una Activity "wrapea" a la transacción justo antes de que ésta comience su

travesía por el workflow, precisamente para incorporar el comportamiento necesario para que la transacción pueda ingresar al workflow. El comportamiento que incorpora el Activity está referido al manejo de excepciones, reglas, historia, etc.

De esta manera, la transacción logra incorporar el comportamiento necesario para poder viajar por el workflow en tiempo de ejecución, esto es, en forma totalmente dinámica, lo cual posibilita que no tengamos que modificar la clase Transaction para agregar nuevo comportamiento, el cual solo es necesario durante el paso de la transacción por este workflow en particular. Aún más, nuestro diseño está orientado a un workflow de las características de MQSeries, pero podríamos necesitar que una transacción tenga que viajar por otro tipo de workflow, donde el comportamiento que se necesita incorporar para posibilitar este viaje, sea completamente diferente al que nosotros especificamos cuando modelamos nuestro workflow. Si tuviéramos que actualizar la clase Transaction por cada tipo diferente de workflow por el cual una transacción pueda eventualmente viajar, nuestro sistema sería muy costoso de mantener. En cambio, si utilizamos Decorators, la responsabilidad de habilitar a una transacción para que viaje por un workflow en particular queda del lado del workflow, que es donde debe estar.

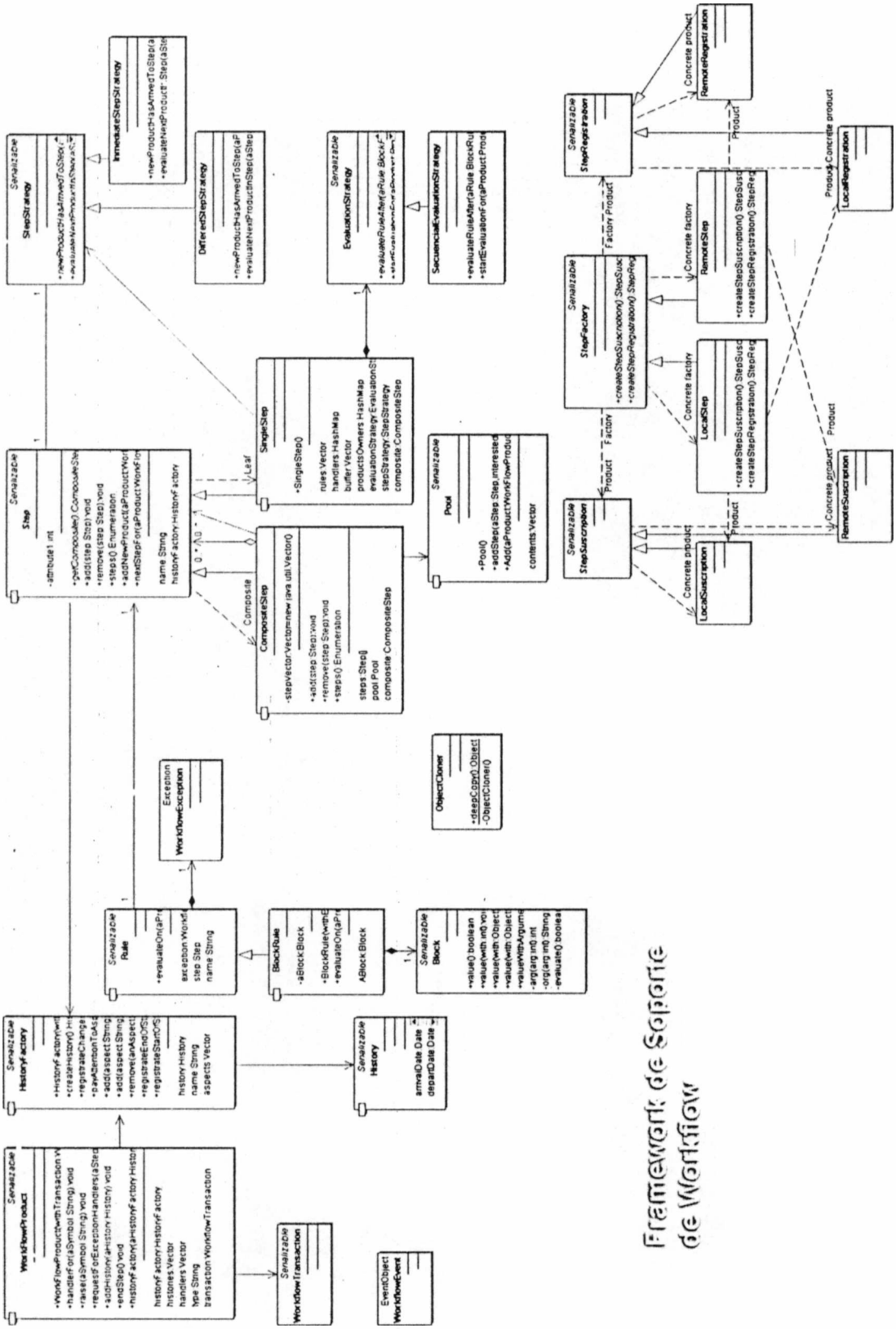


Figura 23. Diagrama general de clases del Framework de soporte de Workflow.

7 Patrones de Diseño para Hipermedia

7.1 Introducción

Los patrones de diseño son utilizados en el diseño de software como un mecanismo para almacenar experiencia. Los patrones describen problemas que ocurren repetidamente y especifican el centro de la solución para dicho problema de tal manera que sea posible reutilizar la solución numerosas veces en diferentes contextos y aplicaciones.

En el campo de orientación a objetos, los patrones muestran micro arquitecturas que van más allá de soluciones simples para problemas recurrentes de diseño. En esta área los elementos importantes de un patrón son las responsabilidades que deben ser asignadas a cada componente (o clase), y las colaboraciones entre ellas.

La importancia de conocer estos patrones es que logramos aprender que hacen los diseñadores experimentados cuando se enfrentan con un problema no trivial.

La utilización de patrones en un diseño orientado a objetos es la clave para maximizar el reuso:

"*Los patrones permiten una amplia reutilización de arquitecturas de software*". La reutilización de arquitecturas es más valiosa que la de algoritmos o estructuras de datos.

"*Los patrones mejoran la comunicación entre y a través de los equipos de desarrollo, ya que proveen un vocabulario común y conciso.*" La utilización de patrones permite que el universo de discusión de los desarrolladores sea de más alto nivel.

"*Los patrones permiten capturar explícitamente el conocimiento que los diseñadores utilizan implícitamente*". Si bien los diseñadores expertos pueden tomar las decisiones correctas de diseño, no siempre documentan el problema que resuelven y las diversas tensiones de diseño en juego que un determinado diseño equilibra.

7.2 Descubriendo Patrones en el Web

Durante los últimos 4 años se ha desarrollado una búsqueda y estudio del uso de patrones en el campo de la hipermedia y hemos estudiado como utilizarlos para recoger la experiencia de los diseñadores de hipermedia. Los patrones en este catálogo están organizados en una taxonomía que incluye patrones arquitecturales, navegacionales, interfaz, mecanismos de búsqueda, comercio electrónico y procesos. Los patrones arquitecturales tratan el problema de descomponer la aplicación Web en componentes. Los patrones navegacionales ayudan a organizar el hiperespacio de tal forma que sea fácilmente navegado. Los patrones de interfaz brindan guidelines para definir objetos que simplifiquen la interacción de los usuarios. Los patrones de búsqueda tratan con los problemas de implementar herramientas de búsquedas en un website. Los patrones de comercio electrónico aportan una mirada detallada a las estrategias de implementación en los sitios comerciales exitosos. Finalmente los

patrones de procesos son la última etapa de nuestro trabajo, donde nos hemos abocado al estudio de implementación de procesos y workflows en aplicaciones Web.

Una buena fuente para encontrar patrones es la Web misma. Mediante el análisis de sitios y aplicaciones, encuadrándolos desde el punto de vista estructural en el modelo OOHDM, hemos encontrado muchas similitudes y las hemos descrito utilizando la notación de patrones de software. Los patrones presentados en el siguiente catálogo son acompañados de numerosos ejemplos disponibles en su gran mayoría en la Web, para su fácil identificación.

7.3 Patrones + Diseño: de los patrones a las aplicaciones.

Como explicamos previamente, los patrones de diseño son útiles para registrar temas recurrentes del diseño en sistemas de software. El descubrimiento de patrones generalmente proviene del estudio de aplicaciones exitosas y la reflexión acerca de cómo solucionan problemas recurrentes los descubre.

Con un catálogo de patrones, un diseñador puede trabajar en un nivel más alto de abstracción porque no necesita solucionar cada problema desde cero, sino aplicar experiencia existente a la situación determinada que está haciendo frente.

Sin embargo, integrar el uso de los patrones de diseño en el ciclo de vida del desarrollo del software no es sencillo. En algunos dominios es posible definir un lenguaje de patrones, es decir, un conjunto de los patrones que no solamente abarcan el conjunto del conjunto de los problemas del diseño sino también se relacionan y proveen una guía de consulta durante el ciclo de vida del desarrollo [Meszaros96].

Hemos desarrollado el método de diseño de Object-Oriented Hypermedia (OOHDM) y lo hemos utilizado para diseñar y para poner diversas aplicaciones del Web en ejecución [OOHDMWeb]; como resultado de nuestra experiencia del diseño, hemos incorporado algunos patrones en nuestro método.

7.4 Incorporación de los patrones en el diseño de la aplicación.

Hay dos maneras de incorporar patrones de diseño en el proceso de diseñar y de poner aplicaciones del Web en ejecución. El primero es enriqueciendo los métodos existentes con primitivas de más alto nivel que permitan la expresión de algunos patrones con una sola notación. El otro es utilizar patrones como guías de consulta activas durante ña tarea de diseño.

Debido a la misma naturaleza de patrones, para cada dominio de aplicación el conjunto de patrones aplicables puede variar. Su introducción tiene un gran impacto dentro de todo proceso de diseño, proporcionando un marco de esquemas buenos y probados de diseño mas que un conjunto de restricciones, mientras que la libertad de la innovación se puede ampliar sobre una base más fuerte. Varias preguntas se presentan en este momento:

¿Cuando entran en juego los patrones?

¿Como se aplican lo patrones en el diseño?

¿Todos los patrones deberían ser aplicados?

El proceso de aplicación de patterns en un diseño no es un proceso aditivo, en el cual piezas preformadas se combinan para crear un conjunto, sino un proceso de desarrollo incremental que va gradualmente, un patrón a la vez. En cada paso un patrón de diseño es aplicado de una manera determinada generando diversas influencias en el proceso de creación y aplicación de otros patrones también.

Llegamos al hecho de que hay dos pautas generales sobre la manera en que un lenguaje de patrones trabaja. El primero se refiere a la importancia del orden de los patrones en el lenguaje, y a la forma (o intensidad) en se aplican los patrones en la secuencia del diseño. Cada patrón trabaja como operador que ayuda a crear y definir aspectos sobre el producto de especificaciones o de aplicaciones anteriores de otros patrones.

El orden en el cual los patrones se hallan organizados dentro de un lenguaje de patrones esta dado por su importancia morfológica o estructural. De esta forma, el lenguaje garantiza que el diseño del sistema completo sea sucesivamente diferenciado.

A menudo, las serias restricciones impuestas por la cantidad limitada de información mostrable, conduce para dispersar equivocadamente la información sobre la base de las "pantallas" y subordinar la modelización conceptual (sea oo o relacional) para reflejar este esquema. Otro acercamiento es encontrar una solución de diseño de interfaz como la "información a pedido", sin tocar el modelo conceptual patrones conceptual. En este caso, un patrón de diseño se aplica después de un cierto diseño para resolver un conflicto común.

Los ejemplos de la interacción entre los patrones y el problema presentado arriba pueden conducir a la creencia que el proceso creativo es "ciego" respecto de los requerimientos entre las etapas de modelado, procurando encontrar el patrón adecuado para cada situación. Eso requeriría encontrar la clausura completa de los patrones de diseño, lo cual es un absurdo puesto que su existencia es el resultado de soluciones probadas y exitosas. Por lo tanto, está claro que dentro del conjunto iterativo de las tareas de modelado y la consiguiente comprensión más profunda del dominio del modelo, el diseño puede plantear nuevos problemas y así el uso de otros patrones a medida que aparecen los nuevos requisitos. Como consecuencia de esto, no es posible establecer de antemano cuales patrones deben ser utilizados dado que su adopción es gobernada por la actividad creativa (que alternadamente, se pudo influenciar por los patrones aplicados anteriormente).

8 Patrones de Navegación

8.1 Navegación Basada en Conjunto (Set Based Navigation)

Intención

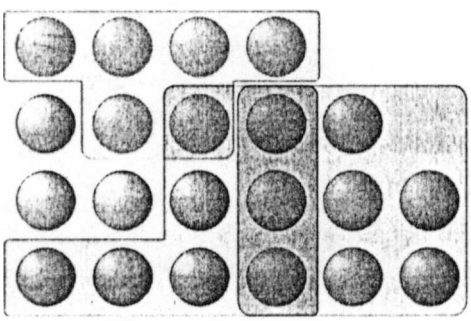
Ofrecer al usuario los sub-espacios de navegación cerrados que son fácilmente navegados.

Problema

Las aplicaciones hipermedia usualmente involucran el tratar con conjuntos de nodos (por ejemplo, Pinturas, Ciudades, Personas, etc.). Puede explorarse estos conjuntos de dos formas diferentes, según la tarea que el usuario esté realizando. Por ejemplo, podemos querer explorar las Pinturas de un Pintor, las Pinturas sobre un tema, etc., y es preferible permitir al usuario moverse fácilmente de nodo en nodo. Ya que los links entre miembros conjuntos generalmente no refleja las relaciones semánticas, raramente son implementados en aplicaciones hipermedia. La estrategia usual seguida por los diseñadores consiste en ofrecer un índice a los miembros conjuntos (ver por ejemplo, los resultados de un Netscape Search); los usuarios deben entonces volver al índice para navegar al siguiente miembro del conjunto. Por lo tanto, se pierde la simplicidad usual de la metáfora de navegación de hipertexto.

Solución

Considere la navegación basada en set como una estrategia de navegación de "primera clase". Agrupe los nodos en conjuntos con significado y ofrezca facilidades de navegación dentro del conjunto, como índices y links para permitir al usuario navegar a los elementos "siguientes" y "previos" al actual. Cuando sea significativo, ofrezca acceso al primero elemento del conjunto. Llamamos a estos conjuntos navegables "contextos de navegación". En la figura 1, mostramos una vista esquemática de un contexto de navegación.



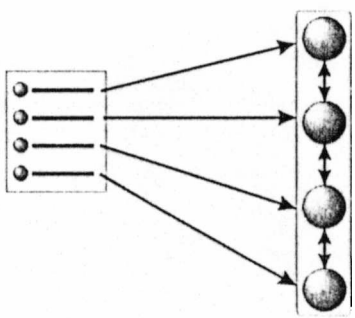


Figura 24. Ejemplo de varios contextos sobre un conjunto de nodos.

Figura 25. Ejemplo de Navegación basada en Conjunto.

La navegación dentro de contextos complementa los links semánticos convencionales, como por ejemplo, aquellos que conectan a una Pintura con su Pintor o Pintores con sus biografías. En otras palabras, el lector puede 'browse' en el conjunto o dejarlo para explorar otros nodos (o eventualmente otros conjuntos). Hay muchas formas diferentes de definir contextos:

Basados en la clase: Los objetos en este tipo de contexto pertenecen a la misma clase C y son elegidos dando una propiedad P que debe ser satisfecha por todos sus elementos; por ejemplo "todas las Pinturas sobre la naturaleza".

Basados en el link: Los objetos en este tipo de contexto son de la misma clase y son elegidos cuando pertenecen a un link 1-a-n. Por ejemplo, "todas las Pinturas de Picasso". Nótese que un caso particular de este tipo es el contexto formado por todos los elementos que son parte de un objeto compuesto.

Enumerados: En este tipo de contexto, los elementos son explícitamente enumerados, y pueden pertenecer a clases diferentes. Un ejemplo típico es un Tour Guiado, como el que muestra en un Museo diferentes Obras de Arte.

Dinámicos: En este tipo de contexto, los elementos son insertados durante la navegación. Un ejemplo de este tipo de contexto es la "historia" que mantienen muchos browsers.

Consecuencias

Establece requerimientos de modelización de la información

Mejora la semántica de la hipermedia. En particular, refuerza la noción original de los hipertextos, con la posibilidad de expresar la semántica adecuada en un determinado contexto y aún asi mantener la noción de unicidad, versus la diseminación de toda esta información en piezas disconexas.

Usos conocidos:

El Contexto de Navegación ha sido usado en muchas aplicaciones hipermedia exitosas. Por ejemplo en la Art Gallery de Microsoft, las pinturas de la London National Gallery son presentadas en diferentes categorías: por lugar, período y tema. Las pinturas en estas categorías pueden ser 'browsed' secuencialmente como se muestra en la Figura 26 y Figura 27.

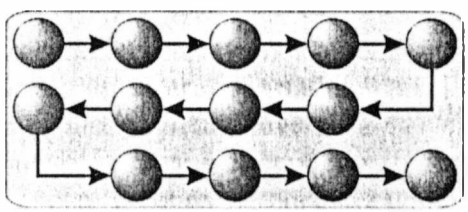


Figura 26. Ejemplo de navegación en secuencia.

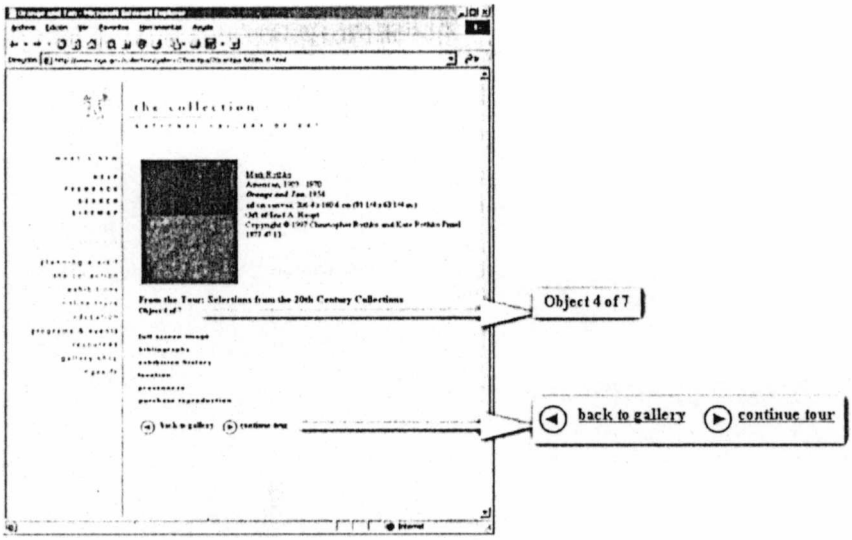


Figura 27. Ejemplo de navegación basada en conjunto en www.nga.gov

8.2 Nodos en Contexto (Node In Context)

Intención

Permitir que el mismo nodo aparezca en diferentes contextos de navegación modificando su apariencia y conexiones con otros nodos según su contexto actual.

Problema

Supongamos que estamos usando la navegación basada en contexto en una galería de pinturas. Cuando llegamos a "Van Gogh", elegimos navegar por sus pinturas, y luego llegamos a "Girasoles". Sin embargo, también podemos llegar a "Girasoles" cuando estamos explorando Pinturas sobre la naturaleza. Es claro que exploraremos el mismo objeto bajo dos perspectivas diferentes. Por ejemplo, cuando se accede a ella como un trabajo de Van Gogh, nos gustaría leer ciertos comentarios sobre su relación con otras pinturas de Van Gogh; también nos gustaría acceder fácilmente a otras pinturas que él haya pintado. Mientras que, como pintura sobre la naturaleza, sería bueno leer (o ver) algo sobre el tema y ser capaz de acceder a otras pinturas sobre el mismo tema (quizá no de Van Gogh). Esto significa que necesitaremos no sólo presentar la información de forma diferente en ambos casos, sino también ofrecer diferentes links o índices.

Nótese que necesitamos acceder a los mismos nodos hipermedia bajo diferentes contextos; estos nodos deberían estar conectados con links (contextuales), por ejemplo, sería deseable que uno pudiera moverse a la "siguiente" pintura en las mismas colecciones. Sin embargo, aunque el contexto en el cual se accede a un nodo no es tema del nodo, es poco práctico y podría esconder inconsistencias tener diferentes objetos para representar el mismo componente pero en cada contexto diferente.

Solución

Desacoplar los objetos de navegación del contexto en que deben ser explorados, y definir las peculiaridades de los objetos como Decoradores [Gamma95], que enriquecen la interfaz de navegación cuando el objeto es visitado en ese contexto. De esta forma los Nodos en Contexto permiten que el mismo objeto pueda ofrecer diferente información, incluyendo links, según el contexto en el cual se está accediendo a él.

En la Galería de Arte, por ejemplo, las anclas que se muestran en la figura 2 permiten navegar a través del contexto actual; por ejemplo, si accedemos a 'Girasoles' como pintura sobre la naturaleza el botón "siguiente" nos permite ver otra pintura sobre la naturaleza. Mientras que si accedemos a 'Girasoles' como pintura de Van Gogh, el significado de "siguiente" cambia y nos permite navegar a otra de sus pinturas. La información contextual se presenta al lectora en el panel central inferior.

Los patterns Navigational Contexts muestran una forma de organizar la estructura de navegación de la aplicación. Los Contextos de Navegación están compuestos de un conjunto de Nodos (como Pinturas) y Links de Contexto (links que conectan objetos en un contexto). Los nodos son decorados con información adicional sobre un contexto particular y anclas adicionales para los links de contexto (usando Nodos en Contexto). Un nodo índice puede ofrecer links a todos los nodos en el contexto o un link al primero.

Volviendo al ejemplo de "Van Gogh", podemos pensar en 'Pintura' y 'Autor' como tipos de nodos de aplicación. Tendremos dos contextos de navegación posibles: 'Contexto relacionado al tema' y 'Contexto relacionado al autor'. Según esto, definiremos decoradores para uno y otro definiendo la información pertinente que mostrará una pintura en cada contexto, y las anclas para la pintura 'previa' o 'siguiente' en el mismo contexto. La figura 3 muestra un diagrama de los elementos que interactúan, donde el nodo de contexto tiene un Índice a los nodos y cada decorador ofrece un ancla al nodo 'siguiente' en el contexto.

Consecuencias

Mayor complejidad de implementación.

Mejor navegación, ya que evita la navegación en estrella, evitando la necesidad de volver al índice para recorrer un conjunto de nodos, puesto que provee la posibilidad de navegar hacia los nodos anteriores y posteriores en cualquier momento.

Evita la repetición de información. En muchos casos, ciertos conceptos pueden tener disponible información extra relevante en un determinado contexto. Nodos en Contexto permite resolver la tensión entre la noción hipertextual de nodo como unidad, y la necesidad de mostrar parte de la información en indeterminado momento.

Usos conocidos

Como se mencionó anteriormente, la Galería de Arte de Microsoft usa ambos patterns de Contexto de Navegación. También pueden encontrarse en Portinari Project [Lanzelotte93] y en "The Way Things Work" ("Cómo Funcionan las Cosas") de Dorling Kindersley.

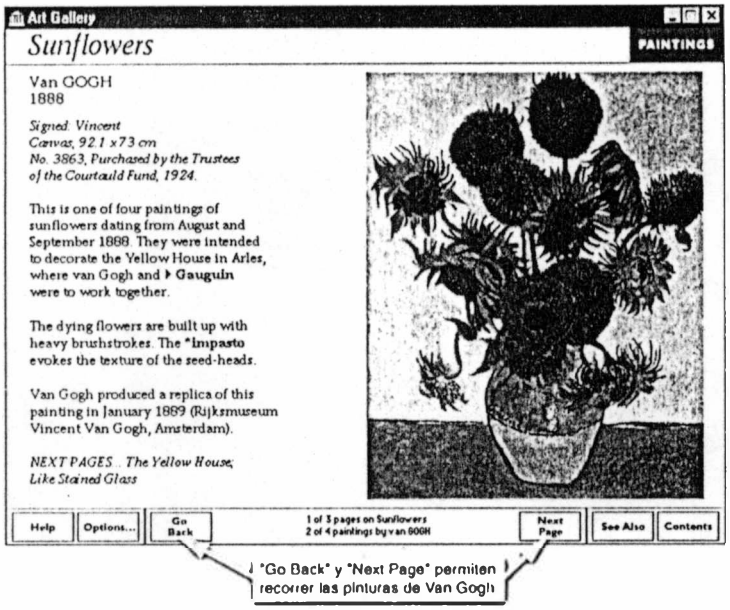


Figura 28. Navegación secuencial a través de las pinturas de MS Art Gallery.

8.3 Referencia Activa (Active Reference)

Intención

Ofrecer una referencia perceptible y permanente sobre el estado actual de navegación, combinando una herramienta de orientación con una forma fácil de navegar a un conjunto de nodos relacionados, en el mismo nivel de abstracción o más alto.

Problema

En muchas aplicaciones hipermedia (particularmente aquellas que involucran estructuras de tiempo o espacio), necesitamos ofrecerle al lector una forma de entender dónde está y ayudarlo a decidir dónde ir a continuación. Por ejemplo en un Museo digital, quisiéramos que el lector vea las obras de arte y a la vez saber en qué lugar del museo está. La solución usual algo ingenua incluiría un índice (u otra estructura de acceso) a los elementos que queremos que el usuario vea. Sin embargo, esta solución requerirá que el usuario vuelva atrás desde el nodo actual al índice para ver dónde está y moverse a otro nodo, asegurando que su posición actual esté subrayada en el índice. Estas operaciones de navegación: volver atrás al índice y luego hacia el otro objetivo pueden desorientar al lector.

Solución

Una buena solución es mantener un objeto de navegación activo y perceptible como índice para otros objetos de navegación (ya sea nodos o subíndices). Este objeto permanece perceptible junto con objetos 'target', permitiendo al usuario ya sea explorar esos objetos o elegir otro 'target' relacionado. Así, podremos interactuar tanto con el índice como también con los nodos 'target'. Nótese que estamos cambiando sutilmente el estilo de navegación usual en aplicaciones (Web) de hipermedia en las cuales cuando nos alejamos de un nodo (o índice), éste es cerrado y sólo podemos volver a él haciendo un paso atrás desde el actual.

Puede encontrarse un ejemplo que aplica esta solución en el web site City.Net (http://www.city.net). Como vemos en la figura 4, el lector tiene una referencia

permanente sobre dónde está mientras explora las ciudades del mundo (ver 'Location: South America->Argentina->Buenos Aires' a la izquierda).

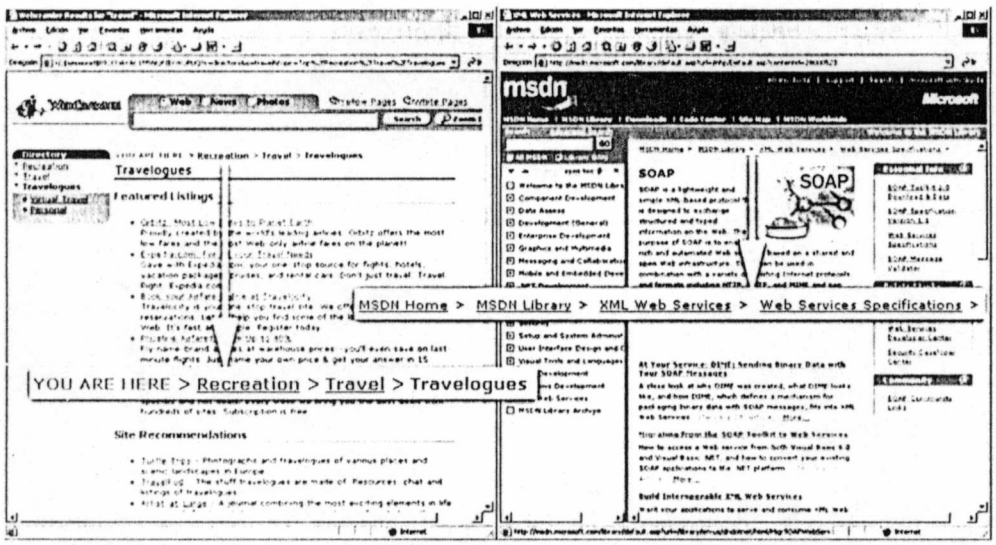


Figura 29. Ejemplos de Referencia Activa en webcrawler.com y msdn.microsoft.com

En particular, puede elegir ir a una región en la cual está situada la ciudad. El patrón se usa sólo parcialmente aquí, mostrando una jerarquía geográfica a la ciudad (incluso si el usuario no visitara todos esos sitios). Una implementación más sofisticada sería ofrecer un índice a todas las otras ciudades de la región. En http://www.citynet.com/, puede verse un índice a todas las otras ciudades cuando se elige una región, aunque la referencia se pierde dentro de una ciudad. Cuando usamos Active Reference, el lector tiene un récord perceptible y permanente sobre el estado de navegación, y así, no sólo ofrecemos una herramienta de orientación, sino también ponemos a disposición los nodos 'target' mientras se navega.

Consecuencias

Mejora la navegación.

Ayuda a disminuir el síndrome de desorientación en el hiper-espacio de información.

Usos conocidos

En "The ultimate Frank Lloyd Wright" de Microsoft, se ha aplicado la Active Reference para mostrar sus edificios en diferentes estados de USA. En Multimedia Beethoven de Microsoft, podemos acceder a una referencia activa de la 9na. Sinfonía. En la Enciclopedia de Grolier, el lector tiene acceso permanente a un diccionario y a nodos a los que se accede desde el diccionario. Este pattern se usa en la web como se vio en el ejemplo anterior (City.Net - http://city.net).

8.4 Novedades (News)

Intención

Dada una aplicación de hipermedia dinámica y grande (por ejemplo un Web-site), ofrecer a los usuarios información sobre los nuevos ítems que se han agregado.

Problema

La mayoría de los web-sites grandes son estructurados como árbol; esta forma de organizar la información ofrece una metáfora de navegación simple, estos espacios de información en general son grandes y casi nunca son completamente navegados por un único usuario.

Supongamos que estamos construyendo un site que ofrece una gran variedad de productos de hardware (como impresoras, escaners, etc.). En este contexto, frecuentemente se puede agregar nueva información. Ya que la mayoría de los usuarios no navegan un web-site ni completamente ni regularmente, este es un tema a considerar, porque el éxito de un producto nuevo puede estar muy relacionado con el conocimiento que el cliente tenga de su existencia. Claramente, información comercial vital como esta no puede ser dejada al azar, a ver si es descubierta. Por otro lado, tratar de solucionar este problema presenta un desafío de diseño para los diseñadores de web, que deben lograr un equilibrio entre un web-site bien estructurado donde la información está organizada en ítems con sub-ítems, etc., y una estructura de navegación con forma de estrella, menos estructurada donde se puede llegar a toda la información a partir de la página 'home'. Este último enfoque claramente es poco deseable porque el uso del site se ve ampliamente reducido y puede volverse inmanejable a medida que crece. Por lo tanto, ¿cómo recibe el usuario feedback instantáneo de cualquier cambio reciente o agregados a la información disponible, y a la vez cómo se mantiene un web-site bien estructurado?

Solución

Estructurar la página 'home' de manera tal que se dedique un espacio a las adiciones más nuevas, presentando "titulares" descriptivos en cuanto a ellas. Usar esos titulares como anclas para conectarlas con páginas relacionadas. Este enfoque permite al diseñador preservar una buena organización de la información, y a la vez da a los usuarios el feedback sobre los cambios que se dan dentro del web-site. Las News (Noticias) implementan atajos a la información que puede ser ubicada en las hojas de un site con estructura de árbol, sin comprometer la estructura subyacente. Nótese que la estructura de navegación de la aplicación se ve levemente afectada por la adición de links temporarios de un nodo a otros.

Los usuarios pueden conseguir información de las noticias entrando al site de la forma habitual. De hecho, a medida que aparecen nuevos titulares, los viejos pueden ser removidos o desplazados. En la figura 6, mostramos 3 web-sites donde se usa el pattern de titulares para anuncios corporativos, http://www.ibm.com, http://www.inprise.com y http://www.sun.com. Una solución más, complementario es agregar un indicador de "nuevo" (un símbolo splash, cambiar el color, etc. . .) junto a los ítems nuevos o recientemente cambiados donde aparecen naturalmente en la aplicación.

Consecuencias

Permite la conciencia(o "awareness") de los usuarios acerca de cambios recientes realizados en el website.

Permite mantener la estructura general del sitio, dando facilidades de acceso mas directo a ciertas partes del sitio.

Usos conocidos

Las News se usan en cientos de web-sites y aplicaciones como http://www.nga.gov donde se usa para anunciar nuevas colecciones o tours actualmente disponibles.

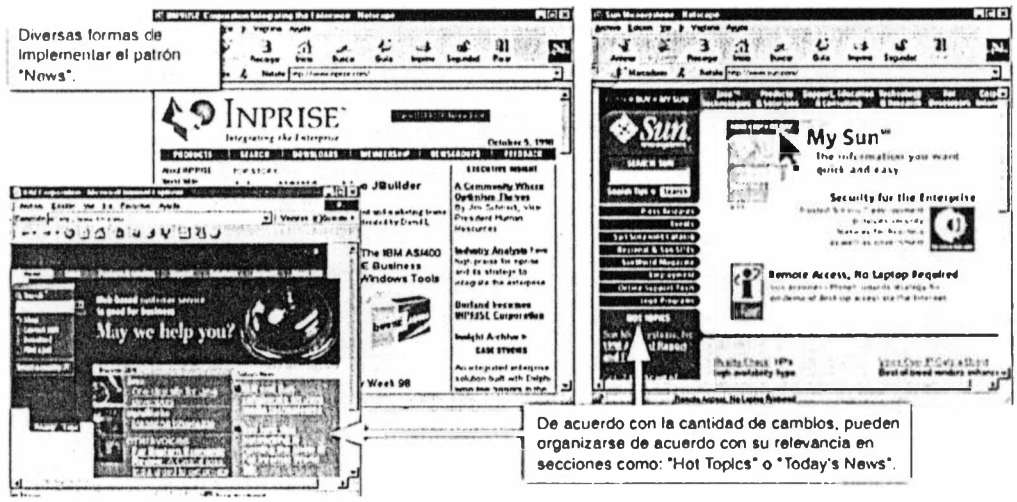


Figura 30. Tres ejemplos del patrón "News", disponibles en inprise.com, ibm.com y sun.com.

8.5 Punto de Referencia (Landmark)

Intención

Ofrecer acceso fácil a subsistemas diferentes aunque relacionados en una aplicación hipermedia.

Problema

Supongamos que estamos construyendo un Sistema Web-Information para un shopping electrónico como www.amazon.com. Al entrar al site, podemos ver muchos productos diferentes como videos, libros o Cds. Podemos explorar los productos y además podemos hacer links a recomendaciones, comentarios sobre el producto, noticias, etc. Cuando describimos el esquema de navegación (es decir, la red de nodos y links), tratamos de seguir de cerca aquellas relaciones existentes en el modelo de objeto subyacente; por ejemplo, podemos navegar de un autor a sus libros, de un CD a la lista de canciones que incluye. Podemos ir de un libro a algunos comentarios que lectores anteriores hicieron, leer sobre libros relacionados, etc. Sin embargo, podemos querer que, en cualquier momento, el lector pueda saltar del almacén de música o libros a su 'shopping basket'. Si construimos el esquema de navegación relacionado cada clase de navegación (como libros, comentarios, noticias, canciones, etc.) al "Music Store", "Book Store" o "Shopping Basket", terminaremos con un esquema difícil de entender que parecerá un plato de fideos; claramente esos links quedan fuera del modelo.

Consecuencias

Mejora la navegación y orientación, proveyendo al usuario de un acceso directo en todo momento a las áreas más importantes del sitio.

Reduce el overhead navegacional.

Solución

Definir un conjunto de landmarks y poder acceder a ellas desde todos los nodos de la red. Hacer que la interfaz de links a un landmark luzca uniforme. De esta forma, los usuarios tendrán una pista visual consistente sobre el landmark. Podemos tener diferentes niveles de landmarking dependiendo el área de hipertexto que estemos visitando. En la figura 7, podemos ver un ejemplo de Landmarks en la librería Amazona.

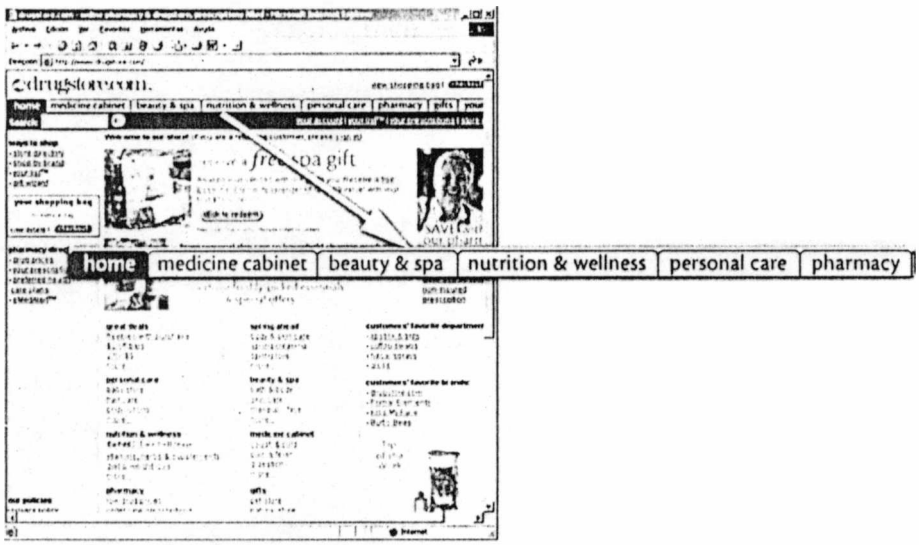


Figura 31. Ejemplo de Landmark en www.drugstore.com

Usos conocidos

Pueden encontrarse landmarks en diferentes aplicaciones Web como www.inprise.com. En "The Way things work" de Macaulay, el lector puede saltar permanentemente a Inventos, Principios de la Ciencia y Mundo de las Máquinas.

8.6 Carro de Compras (Shopping Basket)

Intención

Mantener el control de las elecciones del usuario durante la navegación, haciéndolas persistentes al proceso cuando el usuario lo decida.

Problema

El comercio electrónico es hoy una realidad en la Web. Sin embargo, los usuarios a menudo quieren navegar a través del mercado electrónico para decidir lo que comprarán y en qué momento. En www.amazon.com, por ejemplo, un usuario puede browse a través de cientos de libros o Cds y elegir un sub-conjunto para comprar. La solución naive sería: o pedirle al usuario que compre el producto en el momento que lo encuentra, o forzarlo a marcar todos los productos que le gusta y comprarlos en otra sesión de navegación. Es claro que estos enfoques no se ajustan a los casos en que queremos comprar docenas de productos diferentes, ya que es incómodo para el usuario (e incluso para el negocio) requerir una transacción por cada producto. Esta

alternativa tiene otra desventaja, ya que necesitamos navegar a la página 'check-out' muchas veces y se pierde así tiempo de conexión.

Por otro lado, deberíamos ser conscientes de que estas restricciones al comercio electrónico no deberían interferir con la estructura de navegación de la Web.

Solución

Ofrecer a los usuarios una metáfora similar a las 'marcas', permitiéndole elegir los productos que quiere comprar a medida que los encuentra. Ofrecer un almacén 'persistente' de estos ítems (una shopping basket (bolsa de compras)) a la que se puede acceder como otro objeto de navegación y asociar operaciones de procesamiento como eliminar un ítem, cambiar cantidades, etc.

Esta solución es fácil de implementar agregando un objeto interfaz (usualmente un botón) a cada producto disponible en un lugar de compras. Cuando el usuario elige este producto, es agregado a su lista. Luego, el usuario puede navegar a su 'shopping basket' donde encontrará ya sea una descripción detallada de cada producto o una resumida más un ancla a la página del producto.

En muchos web-sites, la facilidad 'shopping basket' puede ser enriquecida con operaciones de validación. Por ejemplo, si un cliente está planeando sus vacaciones en un servicio de agente de viajes por Internet, el agente de viajes puede ser capaz de chequear tiempos de arribo y salida en reservaciones de hoteles y la viabilidad de conexiones de vuelos.

El resultado es un enfoque muy natural ya que se asemeja a la forma en que la gente realiza sus compras en el supermercado, sumando productos en su carrito mientras habla.

Consecuencias

Provee de una metáfora sencilla para que los usuarios puedan administrar y llevar a cabo una compra.

Usos conocidos

Existen muchos ejemplos de este pattern disponibles en la web. Uno de ellos es Amazon Bookstore (http://www.amazon.com) como se muestra en la Figura 32.

Otro ejemplo es Business travel (http://www.biztravel.com), donde el usuario agrega diferentes destinos a su tour de negocios, junto con reservaciones de auto y hotel. El sistema chequea, entre otras cosas, las fechas de salida y arribo desde diferentes destinos (no puede hacer una reservación de hotel si se va antes de llegar).

Un ejemplo completamente diferente es PublishersDepot (http://www.publishers-depot.com/), un banco de imágenes donde los usuarios pueden buscar diferentes tipos de imágenes (texturas, fondos, etc.) y ponerlos en una lista. Lo que es más, los usuarios pueden crear listas diferentes de dibujos elegidos que son persistentes; la web-site mantiene esas listas de selecciones hechas por un usuario y cada vez que el usuario entra, puede agregar o borrar de esas listas.

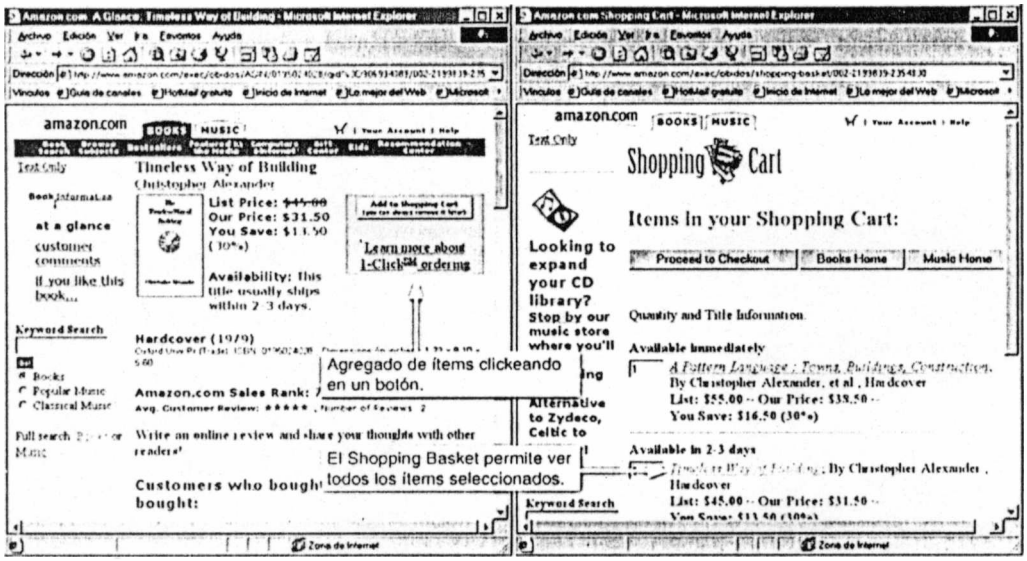


Figura 32. Ejemplo de Shopping Basket en Amazon.com.

Funcionamiento sinérgico de los Patterns de Navegación.

Definir la estructura de navegación de una aplicación hipermedia no requiere sólo definir nodos significativos y conectarlos juiciosamente reflejando las relaciones semánticas en el dominio de la aplicación. También debemos ayudar al usuario a encontrar la información deseada y hacerlo sentir cómodo cuando navega. El ofrecer herramientas de orientación como mapas o atajos como índices es importante, sin embargo usualmente deben solucionarse problemas de organización más sutiles. En este paper, describimos algunos patterns que van más allá del simple modelo de hipertexto de nodos y link y que muestran que pueden mejorarse las estrategias de navegación usuales siguiendo las soluciones en esos patterns.

Los patterns Navigational Context como "Set-based Navigation" permiten ofrecer caminos de navegación a través de miembros de un conjunto, así enriqueciendo la arquitectura de navegación de forma estructurada. Cuando el mismo nodo puede pertenecer a distintos sets, el pattern "Nodes in Context" muestra una forma de desacoplar el nodo del contexto en el cual se accede a él usando el pattern de diseño Decorador. El pattern "Active Reference" muestra una forma de cambiar la semántica de navegación usual de las aplicaciones hipermedia permitiendo a los índices u otras estructuras de acceso coexistir activamente con los nodos 'target'. Cuando la aplicación en sí misma es dividida en varios sub-sistemas, el pattern "Landmark" ofrece una forma para hacer a esos subsistemas accesibles desde cualquier parte de la aplicación. Las "News" y "Shopping Basket" son patterns para los sistemas de información de la Web dinámicos. Mientras que "News" muestra cómo encontrar cosas recientemente agregadas, "Shopping Basket" nos permite hacer una lista de las elecciones de los usuarios para procesarlas juntas cuando el usuario se decida a hacerlo. Este sistema de patterns es complementado con un conjunto de patterns de interfaz que muestra cómo tratar el problema de diseñar interfases multimedia para objetos de navegación en aplicaciones hipermedia.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

9 Patrones de Interfaz

9.1 Introducción

Existe una gran cantidad de buena literatura acerca de los principios de alto nivel del buen diseño de la interfaz, y continúa mejorando a medida que madura esta área. Todos sabemos ahora que se debe utilizar la manipulación directa, el feedback inmediato, affordances apropiados, el uso juicioso del sonido y la animación, protección contra los errores accidentales, mensajes de error apacibles, etcétera. Pero en el caso del diseñador principiante, ya es difícil recordar todos estos principios, mucho mas aún mas el utilizarlos con eficacia.

El diseño de interfaces visuales de la alta calidad para aplicaciones hipermedia es difícil; implica ordenar diversas clases de objetos de interfaz (por ejemplo los que accionan la navegación), prevenir la sobrecarga cognitiva, etc. Desafortunadamente, los métodos de diseño del interfaz no capturan las decisiones o análisis razonado del diseño, así que es muy complicada la registración y comunicación de la experticia en diseño de interfaz

Los patrones que se presentan a continuación dan una guía de consulta simple para su uso en aplicaciones concretas. *Information on Demand* ayuda a decidir qué atributos de nodo mostrar, *Information-Interaction Coupling and Decoupling* muestra cómo organizar los objetos del interfaz según sus roles (objetos de entrada o salida información) del papel y *Behavioral Grouping* los organiza por funcionalidad. Luego, *Behavior Anticipation* y *Process Feedback* explican cómo hacer que el usuario sea consciente y obtenga un feedback acerca del objetivo y comportamiento resultante de una acción. Estas acciones pueden revelarse al usuario mediante *Here I Am*.

9.2 Información bajo Demanda (Information On Demand)

Problema

¿Cómo organizar la interfaz de forma tal que podamos hacer que toda la información de un nodo sea perceptible tomando en cuenta aspectos tales como los cognitivos y los estéticos?

Motivación

En general nos encontramos luchando por decidir cómo mostrar los atributos y las anclas en un nodo. Desafortunadamente, la pantalla usualmente es más chica de lo necesario y muchas veces no podemos usar otros medios (como mostrar simultáneamente una imagen y pasar un audio) ya sea por razones tecnológicas o cognitivas. Supongamos, por ejemplo, una aplicación sobre Pinturas; podemos querer mostrar diferentes atributos de una Pintura, como el nombre del Pintor, año, museo, descripción técnica, etc., pero esto es difícil de lograr si queremos maximizar el espacio que dedicamos a la pintura misma.

Otras Razones:

Un nodo tiene una cantidad de información que debe ser percibida por el lector que no se ajusta en una pantalla toda junta, o puede distraer la atención del usuario (por ejemplo, el audio)

El scrolling a menudo puede no ser aceptable, porque el lector no tiene una visión general sobre lo que encontrará en ese nodo; el lector tendrá que 'scrollear' todo para ver si hay algo que le interesa o no.

Particionar el nodo usando diferentes ventanas para presentar la información, y definir links entre estos nuevos nodos también es problemático, porque en nuestro intento por ajustar el diseño con un tema de implementación, podemos contaminar la estructura de navegación de la aplicación en general. El usuario puede tener el impresión de estar tratando con entidades múltiples, puede desorientarse, cuando de hecho está accediendo a otra parte de la misma entidad conceptual.

Solución

Presentar sólo un subconjunto de atributos, los más importantes, y permitir al usuario controlar qué otra información se presenta en la pantalla, ofreciéndole objetos de interfaz activa (por ejemplo, botones). La activación de estos botones no produce navegación; sólo causan que se muestren diferentes atributos del mismo nodo. Esto sólo se ajusta al principio "Lo que ve es lo que necesita". Deben tomarse en cuenta ciertas consideraciones en esta solución: por ejemplo, podemos querer usar el mismo área de la pantalla para mostrar diferentes atributos, podemos incluso elegir algunos atributos y permitirles aparecer juntos en la pantalla. Cuando se trata con otros tipos de atributos de media, debemos analizar la situación cuidadosamente: por ejemplo una grabación de audio no usa la pantalla; sin embargo puede distraer la atención del usuario, por lo que puede ser más sensato darle al usuario la chance de activar/desactivar ese audio.

Consecuencias

Menos superación de la navegación y cambio de contexto: los usuarios encuentran toda la información como una unidad, eligiendo la información a leer.

El 'scrolling' se reduce al mínimo.

No se adecua a la impresión. La website debe ofrecer una versión 'amiga de la impresora' de la página.

En las implementaciones de la web, se requieren browsers con soporte JavaScript y CSS.

Usos conocidos:

La Información a Demanda se usa en Art Gallery de Microsoft, para ofrecer más información sobre una Pintura; en el CD-ROM "Passage to Vietnam" de Against All Odds, se aplica para permitir al usuario leer más información sobre una fotografía. También se usa en "Le Louvre" de RMM.

En la figura 1, mostramos un ejemplo de Información a Demanda en el contexto del CD de Frank Lloyd Wright de Microsoft. En la figura 2 mostramos un ejemplo del mismo pattern en la WWW.

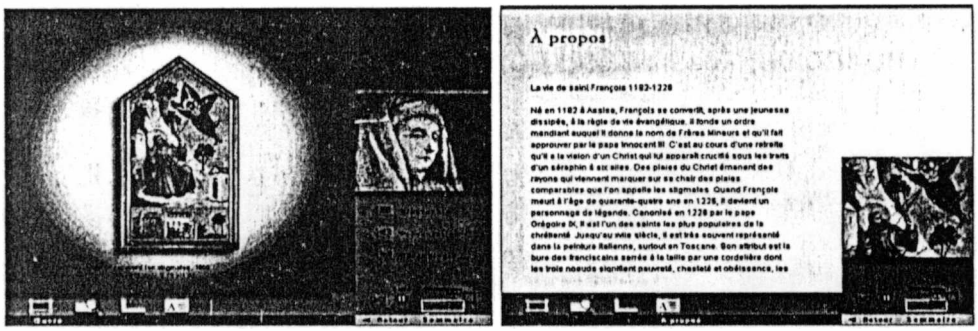


Figura 33. A painting in Le Louvre. The "default" view

Figura 34. Information on Demand in Le Louvre. The icons on the bottom control the presentation of the painting.

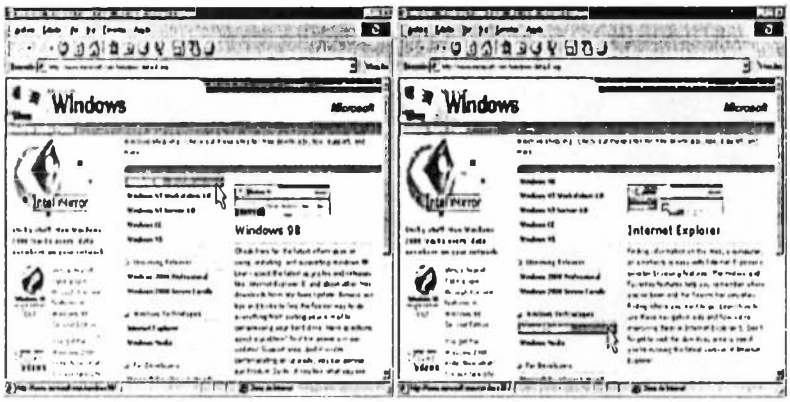


Figura 35 Information on Demand in www.microsoft.com/windows/. Whenever the user selects a different product, the related information is displayed.

Implementación

Existen diferentes plataformas de implementación (Multimedia Toolbook, Macromind/Flash y DHTML -HTML + CSS + JavaScript/VBScript-), sin embargo una guía general de la implementación sería así:

1. Ofrecer una lista de los contenidos de los nodos.
2. Ofrecer un 'handler' para el evento OnMouseClick por cada ítem de la lista.
3. El OnMouseClick handler debería simplemente ocultar el ítem que actualmente se muestra y mostrar el nuevo.

Algunas implementaciones no logran resultados exitosos usando frames, ya que sólo son capaces de ofrecer o navegación en la página, o fuerzan a la navegación cuando se evita el scroll.

9.3 Anticipación de Comportamiento (Behavior Anticipation)

Problema

¿Cómo se le dice al usuario el efecto o consecuencia de activar un objeto de interfaz?

Motivación

Muchas veces, cuando se construye una interfaz, es necesario combinar diferentes elementos de la interfaz como botones, palabras 'hot', controles de medios o incluso controles diseñados 'custom'. Es usual encontrar lectores que se preguntan qué ha pasado después de activar un control, y la consecuencia exacta de la acción realizada.

Solución

Ofrecer feedback sobre el efecto de activar cada elemento de la interfaz. Elegir el tipo de feedback para no ser ambiguo y ser completo: diferentes formas de cursor, iluminación, pequeñas explicaciones basadas en el texto llamadas "tool tips". Además, estos elementos pueden combinarse con sonido y animaciones.

Si estamos usando el pattern de interfaz 'Behavioral Grouping', podemos elegir diferentes tipos de feedback según el tipo de comportamiento que se provee; cuando los controles de interfaz se refieren a un medio en particular, como la animación, podríamos usar un pequeño campo de estado para esa familia.

Consecuencias

Los usuarios tienen información sobre los contenidos del nodo de destino para un ancla dada, así reduciendo la superación de navegación debido a la visita a nodos irrelevantes.

En los sistemas de información de la web, el ofrecer un buen medio para manejar las expectativas del usuario sobre las operaciones que se muestran y su comportamiento real ayuda.

Usos Conocidos

En la figura 3, hay un ejemplo de Atlas Encarta de Microsoft. Cada vez que el usuario posiciona el cursor sobre un elemento de interfaz, aparece un 'tool tip' con una explicación sobre el efecto de activar el control. Se encuentran ejemplos similares en la web, donde los websites usan una combinación Tool Tips GUI standard y JavaScript para mostrar información, como http://www.nervemag.com/ (el ToolTip aparece en la parte de abajo de la página) o JavaScript solamente como la 'homepage' de www.mercedes.com.

Figura 36. Ejemplo de "*Behaviour Anticipation*" del MS Atlas Encarta.

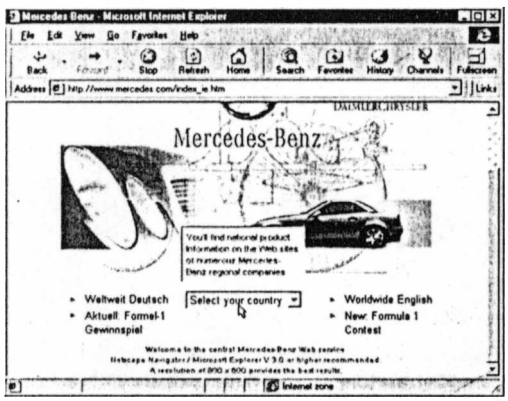


Figura 37

Implementación

La anticipación del comportamiento puede implementarse de forma muy similar al pattern Information on Demand que responde a los eventos mouseOver más que esperar a que el usuario cliquee sobre el ancla, lo que activaría el link. Sin embargo, algunas implementaciones ofrecen una Anticipación de comportamiento audible en su lugar, y se requiere un segundo click para confirmar la activación del link.

En la web, varios websites ofrecen feedback a los usuarios a través de hints GUI standard (ToolTips) usando el modificador ALT del 'tag' <IMG>. Se han hecho otras implementaciones más sofisticadas usando JavaScript para presentar información adicional sobre la barra de estado y animaciones gráficas.

9.4 Desacoplamiento Información-Interacción (Information-Interaction Decoupling)

Problema

¿Cómo se diferencian contenidos y distintos tipos de controles en la interfaz?

Motivación

Una página de una aplicación compleja muestra diferentes contenidos, y se relaciona a muchas otras páginas, así ofreciendo muchas anclas. Lo que es más, si la página ofrece los medios de activación de control más que navegación (como el iniciar alguna 'query'), el usuario puede experimentar la superación cognitiva. Es bien conocido que cuando se ofrecen demasiadas anclas en un texto, el lector se distrae y no puede sacar provecho de todas.

Solución

Separar los canales de comunicación de input de los canales de output, agrupando ambos conjuntos separadamente. Permitir que el 'grupo de interacción de input' permanezca fijo mientras que el 'grupo de output' reaccione dinámicamente a la activación de control. Dentro del grupo de output, también es conveniente diferenciar la 'información substancial' (es decir, contenido) de la 'información de estado'. Esta solución no sólo mejora la percepción de la interfaz de un nodo, sino que también la eficiencia de la implementación.

Consecuencias

Los contenidos son más fáciles de leer, mientras que también se preserva la navegación asociativa entre nodos relacionados.

Algunos websites usan mal esta organización para sobrecargar la página con avisos, promociones, etc.

Figura 38. Ejemplo de Information/Interaction Decoupling en ZDNET. Los links generales están a la izquierda; los links específicos el contenido a la derecha. Nótese que no hay links en el texto mismo.

Usos Conocidos

En la Figura 38, se muestran todos los links a información relacionada sobre el tema actual a la izquierda. Los gráficos/video relevantes al tema se muestra en el medio. Nótese que no hay links en el texto en sí.

Implementación

Aunque el uso del cuadro siempre ha sido algo controvertido como especificación de 'layout', todavía es el enfoque de implementación preferido.

9.5 Agrupamiento por Comportamiento (Behavioral Grouping)

Problema

¿Cómo reconocer los diferentes tipos de controles en la interfaz para que el usuario pueda entenderlos fácilmente?

Motivación

Un problema que enfrentamos a menudo cuando construimos la interfaz de un EIS es cómo organizar los Objetos control (como anclas, botones, etc.) para producir una interfaz significativa. En un EIS típico, existen diferentes tipos de objetos de interfaz activa: aquellos que ofrecen navegación "general", como el botón "back", o anclas para volver a índices, objetos que ofrecen navegación dentro de un contexto; objetos que controlan la interfaz, etc. Incluso cuando aplicamos el desacoplamiento Información-Interacción, pueden existir muchos tipos diferentes de objetos control.

Figura 39. Ejemplo de Behavioral Grouping en http://www.lanacion.com.ar . Observe que la agrupación temática de las diferentes acciones disponibles a los costados laterales y superiores del contenido principal.

Solución

Agrupar los objetos de interfaz de control según su funcionalidad en objetos globales, contextuales, estructurales y de aplicación, y hacer que cada grupo mejore la comprensión.

Consecuencias

Interfases ordenadas, más fáciles de leer.

Ayuda al usuario a identificar operaciones relacionadas.

Usos conocidos

En la figura, se tomó el primer cuadro del CD-ROM "An introduction to programming Java Applets" de MindQ, con grupos de controles de navegación a la izquierda y los controles de mando del tema actual abajo. Se encuentran muchos ejemplos similares en la web (por ejemplo, www.hotmail.com).

9.6 Retroalimentación del Proceso (Process Feedback)

Problema

Cómo mantener al usuario informado sobre el estatus de la interacción de manera tal que sepa qué esperar.

Motivación

Cuando el usuario interactúa con una aplicación hipermedia, puede suceder que muchas opciones resulten en operaciones no-atómicas, tal como contactar otra computadora (en el caso de los browsers WWW) o conseguir información de una base de datos. En tales casos, el usuario puede sentir que no eligió la opción correcta o que él incurrió en una equivocación o inclusive que el sistema no está trabajando correctamente. Esta situación puede empeorar si el usuario se impacienta y comienza a seleccionar la misma opción nuevamente, sin tener conocimiento del encolamiento de pedidos en el que está incurriendo y causando un comportamiento impredecible cuando estas peticiones encoladas sean finalmente.

Solución

Proporcione a un feedback constante sobre el estado de la operación que se está realizando, indicando el avance en el caso de operaciones no-atómicas. Analice cuáles operaciones son atómicas y no necesitan seguimiento. Para las operaciones que son no-atómicas, brindar información sobre el principio, progresión y conclusión de la operación. La clase de feedback a brindar depende del perfil de usuario y de la clase de interacción realizada. Por ejemplo, mientras que para muchas aplicaciones hipermedia (como "Art Gallery" de Microsoft) un cambio de cursor a uno con forma de reloj de arena puede ser suficiente, en las aplicaciones de WWW la clase de feedback es mucho mas elaborado: "buscando servidor", "host contactado", "transfiriendo..." , "Listo.", se visualizaa en la barra del estado; el ícono en la derecha superior del Netscape Navigator Y del MS Internet Explorer muestra animaciones mientras que se está haciendo el proceso.

En browsers de Web, el feedback de proceso es proporcionado directamente por el browser (aunque podemos mejorarlo con la información relacionada al contenido tal como "procesando consulta". Por otra parte, en la mayoría de los ambientes de desarrollo de hipermedia, el autor debe programar el cambio del cursor o definir un área de estado para proporcionar el feedback de la acción que se esta realizando.

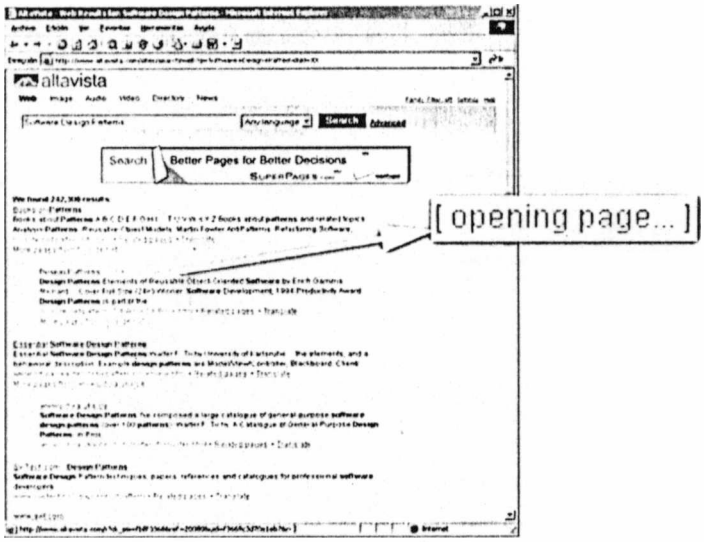


Figura 40. Ejemplo de Process Feedback en www.altavista.com. Nótese que la leyenda "Opening page" aparece cuando se selecciona un link.

Patrones Relacionados

Tanto "Process Feedback" y "Behavior Anticipation" (9.3) tratan un problema similar: como brindar feedback a los usuarios acerca de las acciones que realiza.

Usos Conocidos

En los browsers como Netscape Navigator o MS Explorer, el feedback de proceso se supone para la notificación del estado de la conección http cuando el usuario navega a otra página web. En la mayoría de klas palicaciones CD-ROM, el feedback de proceso es provisto mediante el cambio de la forma del cursor.

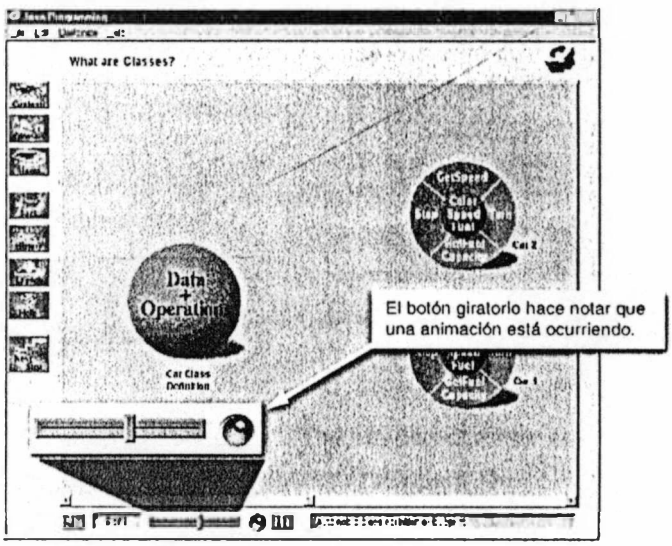


Figura 41. Ejemplo de "*Process Feedback*" Ejemplo tomado del CD-ROM Mind Q's "An Introduction to programming JAVA"

9.7 Aquí Estoy (Here I Am).

Problema.

Cómo mantenemos al usuario informado sobre las acciones que se pueden realizar dentro de uh interfaz compleja o no-estándar?

Motivación.

Cuando un usuario novato interactúa con aplicaciones gráfico-intensivas con una interfaz propietaria, puede ser difícil distinguir los comandos disponibles, los elementos "activos" de la interfaz. Además, cuando varias opciones están disponibles requiere una cierta habilidad en el manejo del mouse para la activación de controles pequeños y links de la hipermedia, lo cual puede conducir a la confusión sobre si se ha seleccionado una acción dada o no.

Solución.

Haga que todos los elementos activos sean sensibles a la presencia del ratón para hacer que se "revelen" al usuario como comando disponible que pueda ser activado si se selecciona en ese momento.

Usos Conocidos.

En las aplicaciones desktop tales como los browsers y en muchas aplicaciones multimedia que presentan interfaces de usuario sofisticadas.

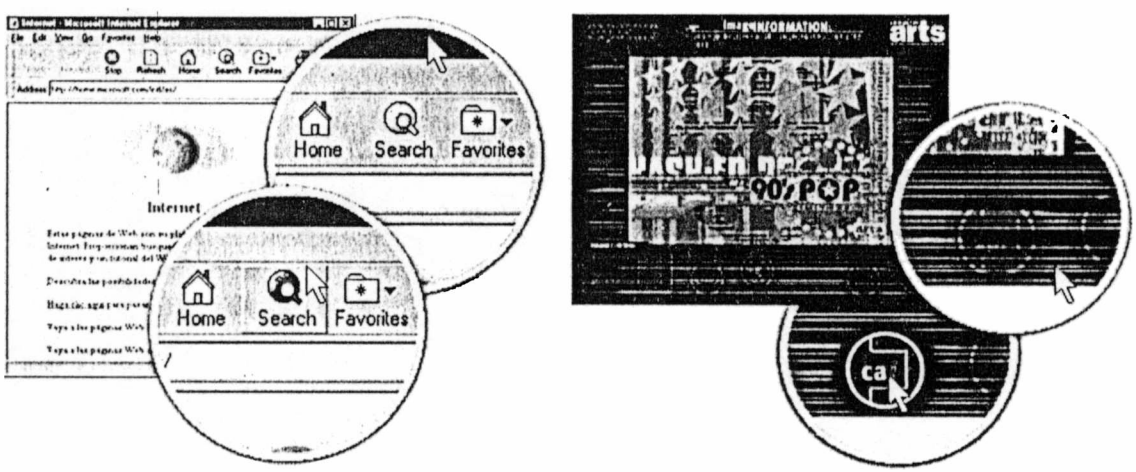


Figure 10: Two examples of "Here I Am". The MS-IE3 browser and the Computer Arts CD-ROM.

10 Patrones de Usabilidad

10.1 URL Affordance

Introducción

URLs son una parte importante de la interfaz usuario de la web porque son el medio de especificar lo que deseamos ver y donde encontrarlo. Como tal, tiene que ser diseñada para que funcione eficazmente. Cuando la gente tiene acceso al Web, ella hace frente generalmente al desafío de encontrar lo que ella desea. Muchas veces, los usuarios saben exactamente lo que están buscando, y tienen cierta confianza de ella existencia, y pueden también saber el direccionamiento del website.

Desafortunadamente, todo este conocimiento no es suficiente para asegurar una búsqueda acertada. Dado que los dominios se están asignando a un ritmo impresionante, encontrar un buen nombre es una tarea ardua.

Mas aún, muchos websites deciden simplificar la selección de nombres y deciden concatenar su nombre conocido en uno solo, formando una cadena larga de caracteres. Esto conduce a URLs de asombrosa longitud o muy difíciles de decir y deletrear.

Además, muchos websites confían en la generación dinámica del contenido y tienden para subestimar el uso de URLs por parte de los usuario para tener acceso a la información que necesitan. En general, estos sitios fuerzan a los usuarios a acceder a su homepage y luego navegar o utilizar una herramienta de la búsqueda para ir al lugar deseado.

Un ejemplo:

http://www.amazon.com/exec/obidos/subst/home/music.HTML/103-1306596-6860655

Sería más fácil de recordar una URL asi:

http://www.amazon.com/music

Esto revela una manera más manejable de expresar un URL. Convierte además a la URL en un mecanismo para tener acceso al homepage del website y también para especificar la información o la actividad específica de interés para el visitante.

Solución

Diseñe la URL como una herramienta complementaria del usuario para encontrar y tener acceso al contenido del website. Lo que sigue es un conjunto de guías de consulta para ayudar al diseñador a alcanzar una usabilidad mejor:

Buscar un nombre de dominio que sea fácil de recordar y deletrear.

a) URLs. breves.

b) URLs fáciles de tipear.

c) URLs persistentes que no cambian para mantener válidos los bookmarks de sus clientes.

d) Utilice mayúsculas o minúsculas en el nombre del dominio, pero nuca ambas al mismo tiempo (eg. NoMezclar.com)

c) Utilice un web server que provea un servicio de spelling-checking para absorber errores tipográficos comunes.

f) Diseñe URLs que ayuden a visualizar la estructura del website.

g) Diseñe URLs que ayuden a visualizar la estructura del sitio.

Existe una heurística sencilla para ayudar a visualizar la estructura del website y su contenido con las URLs:

1. Identifique todas las palabras claves y clases relevantes de los elementos (libros, música, juguetes, etc).
2. Estructure el contenido de su website en diversos directorios, y nómbrelos con las clases de elementos que usted identificó.
3. Fije la página del valor por defecto de cada directorio a una que contenga el índice para esa categoría.
4. Defina los mappings entre palabras claves y elementos, y establezca sus configuraciones de URI del webserver para establecer los redirecciónamientos correctos.

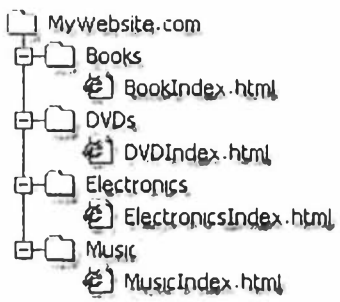


Figura 42. Un ejemplo de como organizar los contenidos de un website

Consecuencias

Positivas

URLs más fácil recordar.

Reduce la cantidad de navegación requerda para tener acceso a una pagina determinada.

Negativas

Fuerza una amplia organización de la jerarquía, basada en la diversidad de opciones. Su desventaja principal es que puede eventualmente representar demasiadas opciones. Mientras que el usuario tiene que clickear solamente una o dos veces para acceder al contenido, el gasto del tiempo con todas las opciones iniciales en la Home Page puede ser contraproducente. Sin embargo, algunos servidores del Web proporcionan a la posibilidad de fijar directorios

virtuales, ayudando a proporcionar URLs sencillas sin forzar una organización específica de la información.

Usos Conocidos

Existen numerosos ejemplos. Una de ellos es Microsoft.com. En su website, Microsoft proporciona a URLs simple para tener acceso a diverso los productos y las tecnologías, cueste lo que cueste su organización real, así proporcionando a conexiones útiles (como www.microsoft.com/sql/) sin fijar limitaciones a la estructura navegacional.

10.2 Versión para Imprimir (Printer Friendly)

Problema

Proveer información a los usuarios de Web en un formato apropiado para ser impresa.

Introducción

El WWW, como implementación del mítico sistema de hipertexto "Xanadu" de Ted Nelson, donde la información estaría siempre disponible como así también todas las versiones de los documentos que contienen tal información, debería haber transformado la necesidad de imprimir documentos en una práctica obsoleta. Lejos de ocurrir eso, dada la incertidumbre de la persistencia de la información a través del tiempo, los usuarios todavía imprimen los documentos disponibles en el Web para ser archivados. Las impresiones son también preferibles si el usuario tiene un sistema basado en el llenado de papeles con carpetas o portapapeles para toda la información relacionada con un proyecto.

Además, se ha demostrado extensamente que la lectura de documentos extensos de texto en la "panta-maños" de una computadora es una tarea desagradable; forzando a los usuarios a imprimir los documentos para su posterior lectura. Finalmente, los usuarios imprimen páginas de la web como folletos en las reuniones o los seminarios [Nielsen99].

Los browsers del Web proporcionan una ayuda limitada para imprimir las páginas del Web, con varias limitaciones al imprimir páginas cuyos contenidos poseen marcos (<frames>). Por ejemplo, la mayoría de los browsers utilizan el mismo tamaño de tipografía y tipo de fuente para su lectura online y la impresión, aun cuando es sabido que ambos medios requieren tipografías diferentes. También la disposición preferida de un documento es diferente: una sola columna para la visualización en línea del documento y disposición a dos columnas para el medio impreso [Nielsen99, Schneiderman94]. Las capacidades del browser dictan cómo será impresa una página web. Sin embargo, hay muchas cosas que pueden realizarse para lograr que las versiones impresas se vean mejor.

Solución

Existen dos estrategias para solucionar este problema, y varias consideraciones en común según las necesidades del diseñador.

Estrategias:

1. Si su sitio tiene varias capas y contiene la información que usted piensa sería útil para recoger y para imprimir de una vez, entonces tome la iniciativa de crear una sola página concatenada.

2. Otro acercamiento para proveer esta funcionalidad es ofrecer una versión descargable y formateada del documento que se imprimirá. Por ejemplo, usted podría ofrecer descargar una versión en Microsoft Word o en PDF del documento entero. De esa manera, usted puede tener control completo del aspecto impreso del documento.

Mejoras Generales:

a) Si usted necesita la disposición y el espaciamiento exactos del documento, utilice tablas. Usted no puede controlar las fuentes que sus audiencias elegirán, el tipo de impresora tienen, o si deciden imprimir en modo del paisaje o de retrato. Si usted realmente necesita controlar el tamaño y el espaciamiento de los bloques de texto, utilice los tablas. Las tablas permiten especificar las posiciones relativas de los bloques y de las imágenes de texto entre si.

b) Preste atención al contraste.

c) La mayoría de la gente imprimirá sus páginas en una impresora PostScript blanco y negro. Las imágenes del color se transforman en el medio-tono blanco y negro, así que es importante que cualquier información crítica en sus imágenes sea de alto contraste. Es decir, si su fondo es oscuro, haga su texto claro y viceversa.

d) NO utilice el fondo como un elemento informativo de la pantalla.

c) Recuerde que las imágenes de fondo no se incluyen en la página impresa.

f) Evite de fijar un tamaño fijo de la fuente en los párrafos o títulos.

g) Recuerde que los usuarios pueden tener resoluciones muy diversas para la visualización. Por lo tanto, el texto que resulta claro de leer en 800x600 puede ser casi invisible en 1024x768 o resoluciones más altas. Los browsers tienen una característica común: tamaño del texto. Esta característica, que permite que el usuario fije la tamaño preferido del texto, llega a ser totalmente inutilizada cuando se establece la tamaño absoluto de la fuente. Es preferible que utilice los tags **<H1...5 > </H1...5 >**.

Consecuencias

Positivas

Los usuarios pueden imprimir documentos en línea sin las limitaciones de representaciones visuales.

Negativas

Mayor complejidad para la construcción de websites que proporcionen tales funciones. Si el website no se diseña correctamente y depende del acoplamiento de la información y presentación (eg. la información se encuentra en formato HTML), hay un riesgo de duplicación de la información para utilizar este recurso.

Ejemplos

Existen docenas de ejemplos disponibles en el web como Borland.com. zdnet.com, sun.com, ibm.com, delphi3000.com

10.3 Acceso Subjetivo (Subjective Access)

Problema

Proporcione a una vista modificada para requisitos particulares del website según el origen de user.s

Introducción

El comercio electrónico ha prosperado en los últimos tres años, y un rango de servicios complementarios se ha construido alrededor de él. Algunos de estos servicios, tales como los de medios de pago, se delegan a otras compañías especializadas, que tienen ya sus propios websites.

Como resultado de esto, los usuarios se desorientan puesto que el pago real para una compra o una subasta se hace en algun sitio probablemente sin relación. Un ejemplo de esto son los sitios de subastas, donde se implementan mecanismos intrincados para asegurar el pago por un lado, y la entregas de la mercancía ofrecida por el otro, se logre de una manera correcta.

Los usuarios internacionales tienen que estableces siempre el lenguaje correcto, aunque el web server sabe ya cuál es el lenguaje preferido. Como resultado de esto, si la ayuda en la lengua preferida nacional no es claramente visible, los usuarios pueden elegir otro website local o más sencillo de operar.

Solución

Provea de los usuarios una vista modificada para requisitos particulares del website, según su precedencia y preferencias, como una transición suave al nuevo website.

Implementación

Para instalar automáticamente el lenguaje preferido, la mayoría de los servidores populares del Web proporcionan ya a estas funciones como característica de estándar. Esta característica aparece generalmente con el nombre de "Content Negotiation". Esto es porque el webserver considera la idea de recursos. Un recurso puede estar disponible en varias diversas representaciones. Por ejemplo, puede ser que esté disponible en diversos lenguajes o diversos tipos de medios, o una combinación. Una forma de seleccionar la opción más apropiada es dar al usuario una página índice, y dejarlo seleccionar. Sin embargo a menudo es posible que el servidor elija automáticamente. Esto funciona porque los browsers pueden enviar como parte de cada petición de información, qué representaciones prefieren. Por ejemplo, un browser podría indicar que quisiera considerar la información en francés si es posible, o inglés en caso contrario. Los browsers indican sus preferencias mediante encabezamientos en cada petición [Apache].

Consecuencias

Positivas

> Los websites puede enfocarse en su negocio, delegando funciones o servicios complementarios a otros websites para lograr una tarea dada, simplificando la configuración entera del website y su mantenimiento.

Negativas

Desorientación potencial de los usuarios, quienes no esperan ser redirigidos, puesto que están acostumbrados al paradigma tradicional de comercio, donde la compra Y el pago se hacen en su totalidad, en el mismo lugar.

Ejemplos

www.ebay.com delegan en www.iescrow.com mucha de las complejidades del pago y de la entrega. El aspecto del website de iescrow es diferente si el usuario esta accediendo desde un link externo a Ebay o si es a través del tipeo de la URL de Ebay en el browser. El otro ejemplo interesante es el webmail de IMP que detecta automáticamente el lenguaje por defecto del usuario.

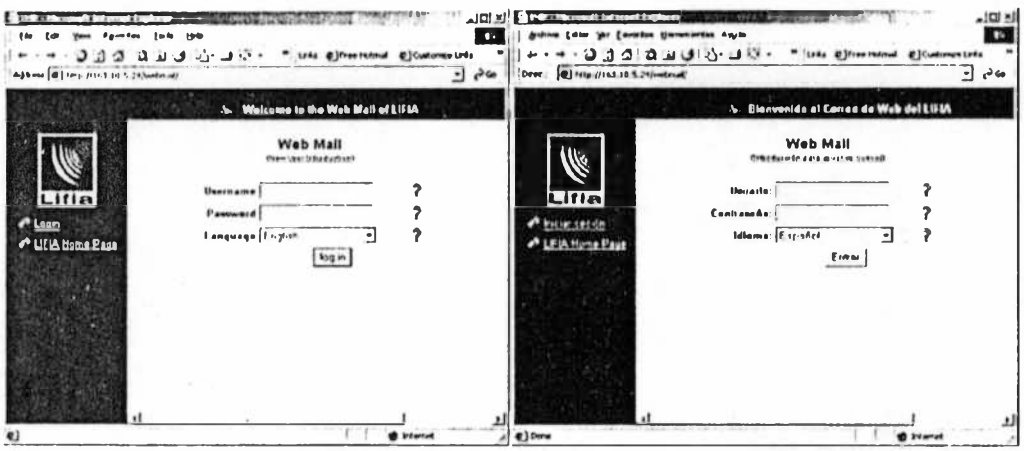


Figura 43. Ejemplo del patrón Subjectivity: el lenguage es automáticamente seteado de acuerdo al lenguaje nativo del browser.

10.4 Link Tipado (Typed Link)

Problema

Ayudar al usuario a distinguir entre diversas clases de links.

Introducción

Las descripciones sobre simplificadas de los links y a donde conducen, pueden hacer que los lectores del hipertexto eviten links que deseamos que sean seguidos, o viceversa.

Las anclas de link deben explicar simultáneamente qué sucederá después de que se sigan y porqué el lector debería seguirlos; este mensaje dual se puede transportar con multivalencia, montaje, o collage.

Desafortunadamente, en el WWW, los links también se utilizan para denotar los objetos que el usuario puede descargar, no navegación. Esto también agrega un nuevo nivel de la confusión puesto que el mismo aspecto visual se utiliza denotar una sección de un documento (conexiones relativas), los documentos en el mismo website o a otro website, y un número arbitrario de clases de objetos como archivos Ms Word, pdf, ps, txt, zip, etc.

Solución

Clasifique la clase de conexiones por el tipo de destino o clase del objeto, y asigne una decoración icónica para proveer del usuario una señal visual.

Ejemplos

Millares de websites sirven de ejemplo de este modelo. La implementación más común es proporcionar a una representación visual del formato del fichero (pdf, txt, zip, los etc.)

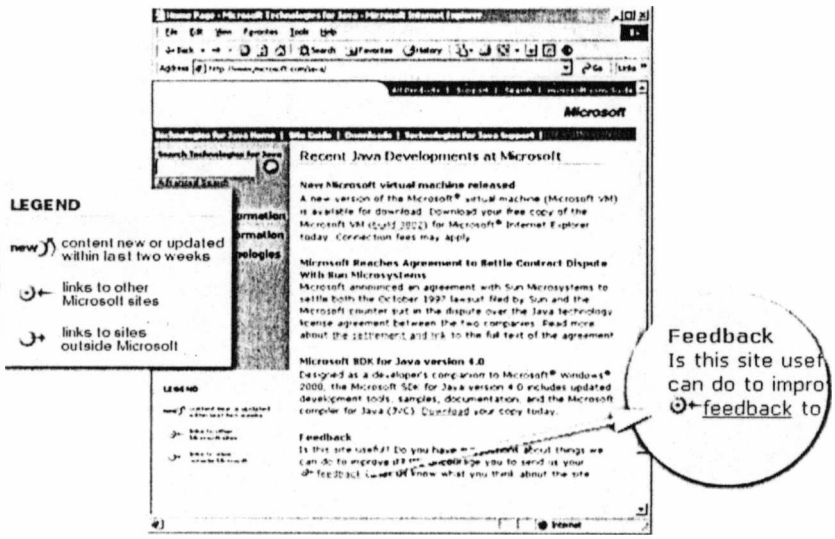


Figura 44. Un ejemplo de Typed Links en Microsoft.com. En este ejemplo, el website define un conjunto de íconos para ayudar a visualizar los links salientes y entrantes.

11 Patrones de Búsqueda

Introducción

Mientras que los métodos de diseño de Hipermedia y del Web han desarrollado y proporcionan ya a las buenas estrategias para el diseño y la organización del espacio subyacente de la información, diseñar las funciones de la búsqueda de una aplicación del Web es difícil y es también una fuente común de ideas falsas extensamente separadas. Algunas de ellas tienen sus raíces en la creencia común que la adición de un servicio de búsqueda está reducida al interfaz entre un motor de la indexación de direcciones y el website. Al proporcionar a las capacidades de la búsqueda para una aplicación del Web, necesitamos entender que las expectativas y las necesidades del utilizador bajen a menudo bien más allá de la búsqueda de palabra clave simple.

Cuando el utilizador comienza a buscar, habrá cierta información en el espacio de la información que será altamente relevante a sus necesidades de información, algunas de las cuales serán útiles, algún marginal relevante y algo no relevante en todos. Solamente algo de esta información relevante será devuelta por un Search Engine. Es posible medir el funcionamiento de una búsqueda usando dos relaciones de transformación: relación de transformación de memoria y relación de transformación de la precisión. La *precisión* es la relación de transformación del número total del documento relevante (o de nodos, de páginas, del etc.) extraído, al número total de los documentos extraídos, mientras que *recuperación* mide la relación de transformación del número de los documentos relevantes extraídos, al número total de documentos relevantes.

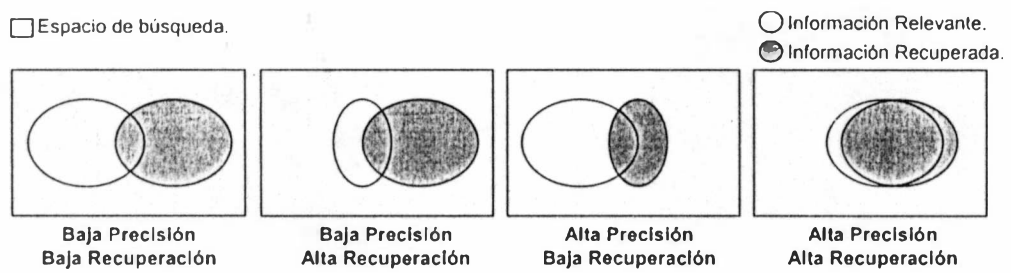


Figura 45. Precisión y Recuperación

En esta sección discutimos algunas ediciones del diseño que un diseñador novato debe conocer, y mostramos cómo los diseñadores expertos las solucionan. Se explicitan las diferentes tensiones de diseño que se deben considerar al agregar capacidades de búsqueda a un sistema de información del Web, mostrando diversas clases de búsqueda. Mostramos que estos problemas del diseño son muy comunes en aplicaciones actuales del Web, ilustrándolas con los sistemas de información comerciales actualmente accesibles.

La mayoría de estas estrategias de la búsqueda se pueden poner en ejecución utilizando tecnología estándar o reutilizando las herramientas existentes de la

búsqueda. No discutimos la implementación de los algoritmos de la búsqueda: esta edición puede ser el foco de un lenguaje complementario del modelo que trata aspectos de la implementación. Los conceptos se muestran en la Figura 45.

11.1 Espacio de Búsqueda Seleccionable (Selectable Search Space)

Intención.

Especificar una categoría en la cual hacer o restringir la búsqueda.

Motivación.

Muchas veces, los usuarios tratan de acceder a la información a través de la navegación. Sin embargo, como los espacios de información son cada vez más grandes, no siempre tienen éxito en la búsqueda de la información deseada. Por lo tanto, habiendo llegado a cierto punto con la navegación, realizan una búsqueda para encontrar la información deseada.

Toda actividad de búsqueda persigue el objetivo de proveer resultados de búsqueda con la mayor precisión y recuperación. También es importante notar que, con la mayoría de las maquinarias de búsqueda, es posible equilibrar precisión y recuperación. Si aumentamos el número de documentos devueltos, es posible que el número de documentos relevantes devueltos también aumente. Sin embargo, al mismo tiempo también es probable que el número de documentos irrelevantes devueltos aumente, así disminuyendo la precisión. En este punto, la tarea de poner a tono una búsqueda puede verse simplificada con un análisis de la información que se publica en el website y cómo está estructurada, para encontrar grupos de documentos/nodos semánticamente relacionados.

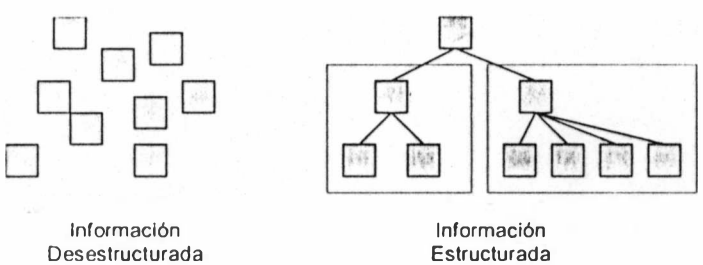


Figura 46. Información Estructurada vs. Desestrucurada

El poder agrupar semánticamente la información permite mejores resultados en las búsquedas.

Muchas compañías ofrecen sus productos en sus websites. Cuando un usuario está buscando algo concreto, es inútil buscar en áreas de productos diferentes a la que pertenece el producto deseado. Supongamos que una persona está interesada en comprar cierto libro, y el website elegido también vende videos y regalos; tendría sentido permitirle al usuario elegir la categoría de productos en la cual realizar su búsqueda. En la Figura 47, podemos ver cómo establecer estas categorías en un servidor de búsqueda como Netscape Enterprise.

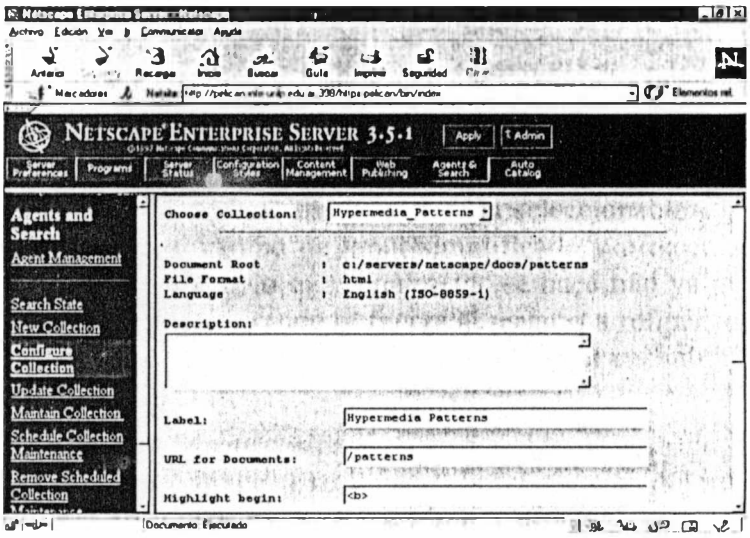


Figura 47. Configuración de categorías para las búsquedas en NES.

Aparece un caso más complejo cuando sería útil combinar espacios de búsqueda o categorías. Considérese un website que ya tiene su información organizada en distintos grupos. Es posible que parte de la información relevante a un tema dado pueda aparecer en más de uno de estos agrupamientos. Se da el caso de muchas compañías que aplican un conocimiento o tecnología dada en varias áreas o productos, y por lo tanto pueden existir reportes sobre distintos aspectos del tema de búsqueda cubiertos en diferentes regiones de los websites o, incluso en otros websites de la misma compañía. En tal contexto, se hace necesaria una mayor flexibilidad, ya que el website ya tiene varias categorías, sería poco práctico considerar todas las combinaciones de áreas posibles.

Razones:

Los usuarios quieren buscar en grandes espacios de búsqueda.

La precisión y recuperación son medidas que se oponen mutuamente, mientras que la búsqueda efectiva debería tener una precisión y un recuperación altos.

La efectividad de la búsqueda puede mejorar restringiendo el espacio de búsqueda a subespacios relevantes conocidos.

Solución.

Ofrecer al usuario un mecanismo para elegir en qué categoría (subespacio) el usuario va a buscar. Hay dos variantes comunes a esta solución, según la funcionalidad deseada:

Permitir a los usuarios elegir sólo una categoría de búsqueda por vez. El requerimiento para implementar esta solución es muy simple: sería imposible dividir el espacio de información en subespacios desarticulados. La misma información no debería pertenecer a más de un grupo por vez.

Permitir a los usuarios combinar áreas de búsqueda. Se le presenta al usuario un agrupamiento de checkboxes, con todas las áreas de búsqueda posibles. El usuario puede elegir cualquier combinación. Aunque este enfoque es más flexible, debería usarse sólo cuando el anterior no es posible, ya que requiere navegación extra a una página de búsqueda para elegir las categorías deseadas.

Consecuencias

Ventajas

Tasas de recuperación y precisión más altas. Sorprendentemente, una gran cantidad de websites con enormes cantidades de información se niegan a ofrecer a sus usuarios espacios de búsqueda seleccionables. Más bien, estos sitios se basan en un campo de input simplificado y ofrecen sólo búsquedas completas en un website, lo que usualmente se hace mal ya que la página de búsqueda es mucho más grande, y fuerza al usuario a refinar repetidamente la búsqueda ofreciendo información más detallada, para encontrar la información buscada.

Se necesitan menos búsquedas para encontrar el tema deseado, así se consume menos recursos de cómputo, y se aumenta la satisfacción del usuario.

Desventajas:

Las categorías usualmente deben ser determinadas manualmente.

El tener demasiadas categorías tiene un impacto sobre su utilidad. Por lo tanto, debe lograrse cierto equilibrio entre la granularidad de los espacios de búsqueda y su cantidad.

Ejemplos.

Se muestran dos ejemplos de Selectable Search Categories en la figura 5. El primero es amazon.com, que implementa espacios seleccionables con un combo-box, mientras que la segunda implementación en microsoft.com usa 'radio-buttons'. Ambas implementaciones permiten elegir sólo una categoría por vez.

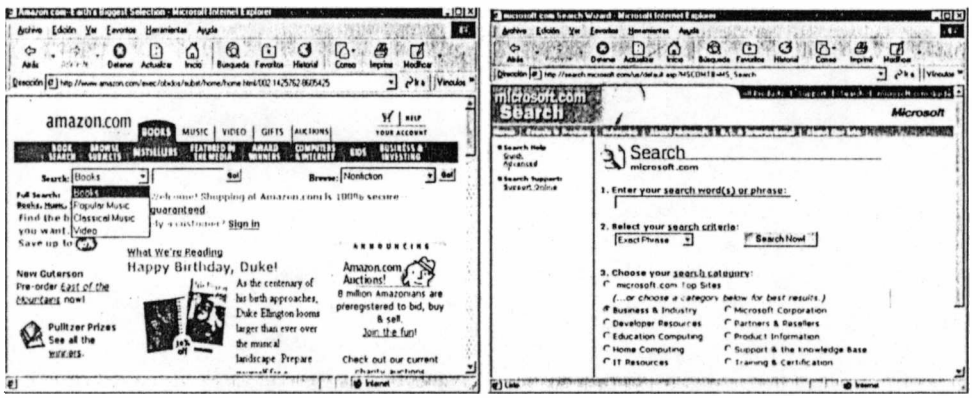


Figura 48. Ejemplo de Selección de Categorías en Microsoft.com

Figura 49. Muestra un ejemplo de áreas de búsqueda combinables en www.microsoft.com.

Patterns Relacionados.

La Búsqueda a Demanda (Search on Demand) a menudo es mejorada con Selectable Search Spaces (Espacios de Búsqueda Seleccionables). También se usa en combinación con Node in Context (Nodo en Contexto) para establecer automáticamente la categoría de búsqueda.

11.2 Palabras clave seleccionables (Selectable Keywords)

Intención.

Sugerir un conjunto de keywords que sean relevantes a los resultados de las búsquedas previas, para mejorar la calidad de futuros resultados de búsqueda.

Motivación.

La interacción más común que los usuarios tienen con las maquinarias de búsqueda es tipeando 'palabras clave' (keywords) relevantes sobre el tema de búsqueda. Sin embargo, ya que el espacio de búsqueda en la web es muy grande, pueden existir muchas posibilidades para una palabra dada, así debe especificarse una búsqueda más sofisticada, que casi siempre requiere más palabras clave para reducir el espacio de búsqueda y permitir un mejor hallazgo.

Razones.

Los usuarios quieren buscar en grandes espacios de búsqueda.

Palabras clave arbitrarias pueden dar como resultado ítems no deseados, mientras que las predefinidas pueden aumentar la efectividad.

Los usuarios no siempre conocen qué palabras claves pueden usar.

Solución.

Ofrecer al usuario una lista de palabras clave posibles, según la búsqueda que ya haya hecho el usuario. Debería poder incluir cualquier combinación de palabras clave sugeridas sin tipearlas, ya que esta es una actividad que lleva a cometer errores.

La estrategia de interfaz usual para esta solución es la siguiente: cerca del campo en el que el usuario entra las palabras claves para realizar la búsqueda, debe haber una lista

de palabras clave, que deberían ser seleccionables y combinables (por ejemplo: usando checkboxes).

Consecuencias.

Positivas

Los usuarios pueden completar una búsqueda sin tipear.

Resultados más exitosos, gracias a mejores 'matches'.

Negativas

Los usuarios deben realizar al menos una búsqueda, antes de que el sistema pueda sugerir una palabra clave.

Ejemplos.

Excite.com ofrece al usuario un conjunto de palabras clave relacionadas a una búsqueda hecha anteriormente, ver figura 7. En la figura 8, vemos una implementación menos flexible en altavista.com.

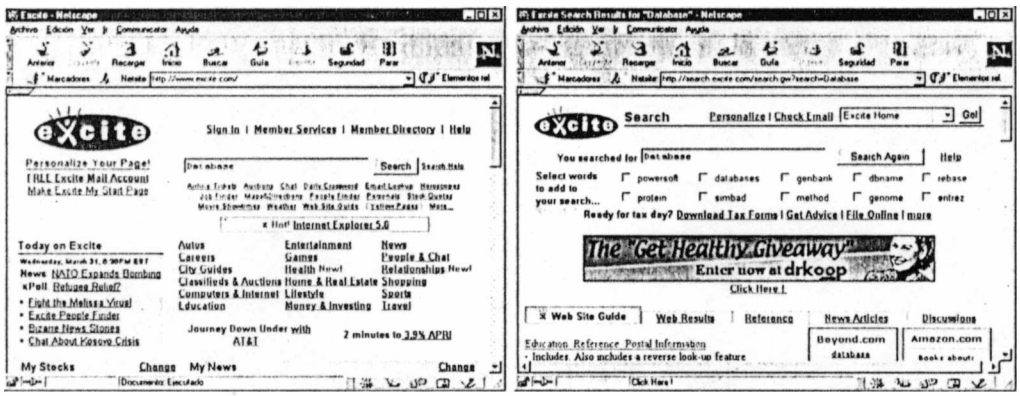


Figura 50. Un ejemplo de Palabras Clave Seleccionables en www.excite.com.

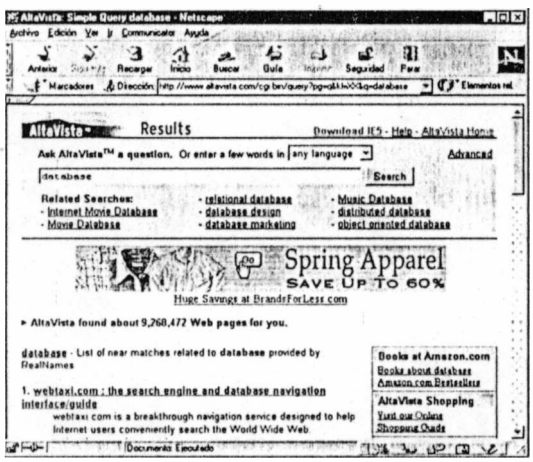


Figura 51. En esta implementación, la máquina de búsqueda propone varias búsquedas pre-calculadas.

Patterns relacionados.

Behavioral grouping [Garrido97]. El campo búsqueda/palabra clave y la lista de temas seleccionables debería conformarse/estar arreglado en una unidad lógica. Otros patterns relacionados son Structured Answer.

11.3 Respuesta Estructurada (Structured Answer)

Intención

Ofrecer al usuario un reporte de información organizado como resultado de una 'query' ('investigación').

Motivación.

El campo de investigación de la Interacción Computadora-Humano ha estudiado durante años la comunicación entre las computadoras y sus usuarios. Uno de los aspectos claves de interactuar con computadoras es el problema de entender un mensaje o resultado, simplemente porque las computadoras no ofrecer en sí información importante. La idea del contexto en el cual se da la respuesta. El contexto es vital en la comunicación humana, para darle al interlocutor la chance de entender correctamente una información dada.

En la gran extensión del espacio de búsqueda de la web, los resultados pueden incluir una gran cantidad de links de información relevante. Sin embargo, como la cantidad de información es enorme, debe ofrecerse una explicación más detallada sobre la naturaleza de la información recibida para permitirle a la gente ubicarse en un contexto para entender la información.

Supongamos por ejemplo, que iniciamos una 'query' usando la palabra clave: "Objetos". En una búsqueda standard, obtendríamos miles de links que apuntarían a distintos tipos de páginas que incluyen la palabra: Objetos. Incluso si usamos Selectable Keywords, la búsqueda podría ser algo chata.

El punto es que una lista de links no es suficiente. Se necesita mayor información para ayudar al usuario a decidirse por el link correcto.

Razones.

Los resultados de las 'queries' pueden incluir muchos ítems.

Se necesita información resumida para ayudar a los usuarios a elegir entre alternativas.

Cuando se examina un ítem, debe disponerse de fácil acceso a otros ítems.

Solución.

Ofrecer al usuario un resultado más estructurado que organice la información para ayudar al usuario a decidir qué opción es la mejor. Puede agruparse tal información adicional en diferentes categorías:

Información sobre el ítem que se ha encontrado: incluye un breve 'abstract' de la página, lo que puede hacer automáticamente una maquinaria de búsqueda.

Información sobre los links a otra información que puede relacionarse pero que es elegida automáticamente por el sistema.

Una taxonomía que organice el espacio de información en subespacios significativos.

Consecuencias

Positivas:

Los usuarios tienen una visión más completa de la información recibida y por lo tanto pueden reducir el esfuerzo de navegación y búsqueda.

Negativas:

Ya que se ofrece información extra, la cantidad de información recibida por un usuario inexperto puede superarlo.

Ejemplos

En www.excite.com y www.yahoo.com, podemos encontrar buenos ejemplos de Respuesta estructurada, donde se ofrece mucha información sobre los temas encontrados como ratio de precisión, categoría, temas relacionados, noticias en cuanto a algunas de las palabras clave, links a libros relacionados y dónde comprarlos. En www.altavista.com, también encontramos una representación gráfica del espacio de resultado, y los temas relacionados. En la figura 9, vemos la solución en www.yahoo.com para la 'query' sobre 'objetos.

Figura 52. Ejemplo en www.yahoo.com

Patterns Relacionados

La Respuesta estructurada juega un doble rol con respecto a las Palabras clave seleccionables. Mientras que estas últimas permiten reducir el espacio de búsqueda, la Respuesta estructura da más información sobre el espacio del resultado, organizando los resultados en diferentes categorías.

11.4 Maquinaria de Búsqueda Seleccionable (Selectable Search Engine)

Intención.

Aumentar las capacidades de búsqueda de un website delegando el servicio de búsqueda real a maquinarias más sofisticadas, que pueden ser capaces de producir una Respuesta Estructurada con información más detallada o más formateada.

Motivación.

La web ofrece un enorme espacio de información que está lejos de estar organizado, y crece a una tasa extremadamente alta, sufre un número de cambios imprevisible, donde la información es agregada o removida, clasificada y reorganizada, con criterios altamente heterogéneos. Para ayudar a los usuarios a encontrar un tema dado

en tal cantidad de información, la web es escaneada continuamente por 'indexers', 'spiders', 'worms', etc., para poder poner al día sus propias bases de dato y ofrecer a los usuarios resultados de búsqueda exactos.

Sin embargo, como la cantidad de información es tan alta, ninguna maquinaria de búsqueda hoy en día puede decir que tiene conocimiento completo de la web, por lo tanto la misma búsqueda realizada en diferentes maquinarias de búsqueda ofrece resultados diferentes. Cuando un 'buscador' no encuentra un tema dado, puede ser necesario repetir la misma tarea con otros. Este es un tema ya conocido para los usuarios experimentados de la web, y como resultado de esto, varias compañías ofrecen en sus 'homepages' la habilidad de realizar una búsqueda usando la maquinaria de búsqueda en la web favorita de los usuarios sin tener que navegar a otros sitios. Incluso 'webbrowsers' como MS-Internet Explorer ya ofrecen este rasgo, como una forma de mejorar la tarea de búsqueda en la web.

Razones.

Las maquinarias de búsqueda poderosas son caras y consumen recursos.

El mantener los índices también consume recursos.

Las tecnologías de búsqueda están evolucionando rápidamente, y puede consumir tiempo mantenerse al día con ellas.

Solución.

Cuando se desarrolla un sistema de información de hipermedia basado en la web, es posible unir una maquinaria de búsqueda ya existente que permita a websites externos usarla. Por esto, es posible agregar capacidades de búsqueda completa a un website delegando la búsqueda a estas maquinarias que permiten ser usadas desde lugares remotos. Por lo tanto, el contexto descripto tiene dos caras: los usuarios pueden elegir maquinarias de búsqueda sin tener que navegar a sus websites respectivos y, por otro lado, también permite implementar una especie de servicio de búsqueda "costo cero" en un website.

Consecuencias

Positivas

> Los tiempos y costos de desarrollo se reducen y podemos confiar en el poder de las maquinarias existentes.

Negativas

> Las maquinarias existentes pueden ser rígidas para la tarea en cuestión. Debemos evaluar la relación costo/beneficio de delegar búsquedas en términos de funcionalidad de nuestra aplicación o sitio de Web.

Ejemplos

Pueden usarse muchas maquinarias de búsqueda de otras aplicaciones. Por ejemplo: Amazon, infoseek, Altavista, Excite, etc.

Puede encontrarse este pattern en muchos websites que implementan Selectable Searchers: NetscapeCenter (Figura 53), MSN (www.msn.com, www.webtaxi.com, etc.) Además, www.uiuc.edu, donde los usuarios pueden buscar en el website del campus usando una maquinaria de búsqueda externa.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

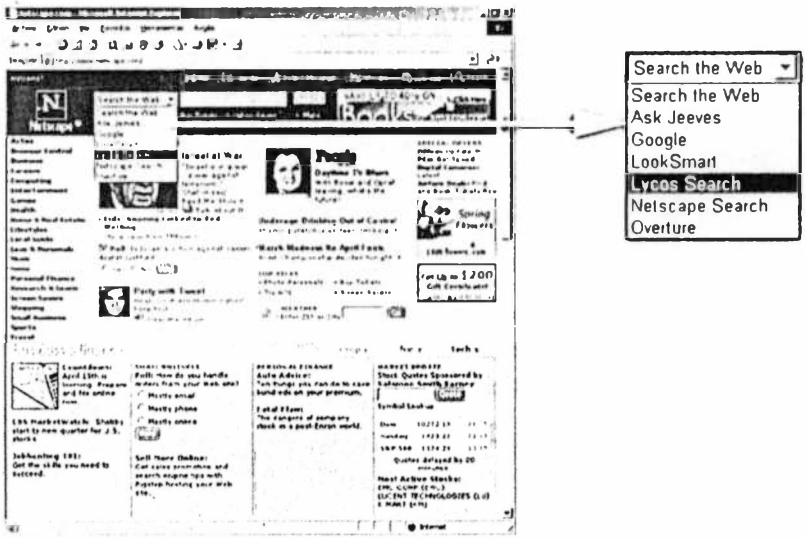


Figura 53. Ejemplo de Maquinaria de Búsqueda Seleccionable.

Patterns Relacionados

Behavioral Grouping [Garrido97], ya que la habilidad para elegir una maquinaria de búsqueda debe ser dada localmente al campo de búsqueda de la palabra clave.

11.5 Interfaz de Búsqueda Sencilla (Simple Search Interface)

Intención.

Ofrecer a los usuarios mecanismos de búsqueda poderosos, pero simples.

Motivación.

Los mecanismos de búsqueda efectivos a menudo involucran formulaciones de 'query' sofisticados y complicados, que usan operadores lógicos, refinamientos de 'query', calificadores de campo, etc... Por otro lado, a menudo una simple 'query' logrará satisfacer las necesidades del usuario. Como ejemplo, consideremos un desarrollador que accede a un website buscando cierta especificación técnica, aunque puede ser posible acceder por la navegación, se hace una búsqueda rápida para ayudar a ir directo a la información requerida.

Aparece un contexto similar cuando se accede a un website para comprar, por ejemplo, un libro. En este último caso, la tarea usual no es navegar, sino buscar, ya que es más natural pensar en la búsqueda de un libro en una librería más que buscar en todas las categorías de todos los libros para encontrar un título dado. En los ejemplos descriptos, existe una característica común entre ambos websites: a menudo se usan las capacidades de búsqueda, o incluso más que navegar para acceder a la información requerida y por lo tanto, si siempre es necesaria cierta navegación extra para acceder a la facilidad de búsqueda, se convierte en una fuente de superación de navegación innecesaria. Lo que es más, a medida que las maquinarias de búsqueda se vuelven más sofisticadas, la cantidad de opciones disponibles al usuario para realizar una búsqueda pueden transformar una actividad abiertamente simple en un esfuerzo más complejo que lleva mucho tiempo.

Razones

Los usuarios ingenuos en general hacen búsquedas simples.

Las búsquedas simples a menudo son suficientes para las necesidades del usuario.

Capacidades de búsqueda avanzadas requieren más experiencia del usuario.

Formular 'queries' avanzadas requiere de más tiempo, y requiere una interfaz más sofisticada que se justifica sólo cuando es realmente necesaria.

Solución

Ofrecer una interfaz de búsqueda directa y muy simple - un campo, sin operadores - esto logrará satisfacer a una gran parte de los usuarios; además ofrecer un link a una interfaz de búsqueda más refinada, donde estén todas "las campanas y silbatos". Ofrecer una forma consistente de moverse de una a otra.

Consecuencias

Positivas

Un acceso más rápido a las capacidades de búsqueda provistas por la website

Menos superación de navegación para 'queries' simples

Negativas

Suma un elemento extra a la interfaz del usuario, que a su vez puede resultar en interfases más complejas para los usuarios inexpertos.

Ejemplos

Hay muchos ejemplos en la Web, como Altavista.com, yahoo.com o softseek.com, que ofrecen una interfaz de búsqueda simple, pero el usuario también tiene opciones de búsqueda más sofisticadas en una página separada.

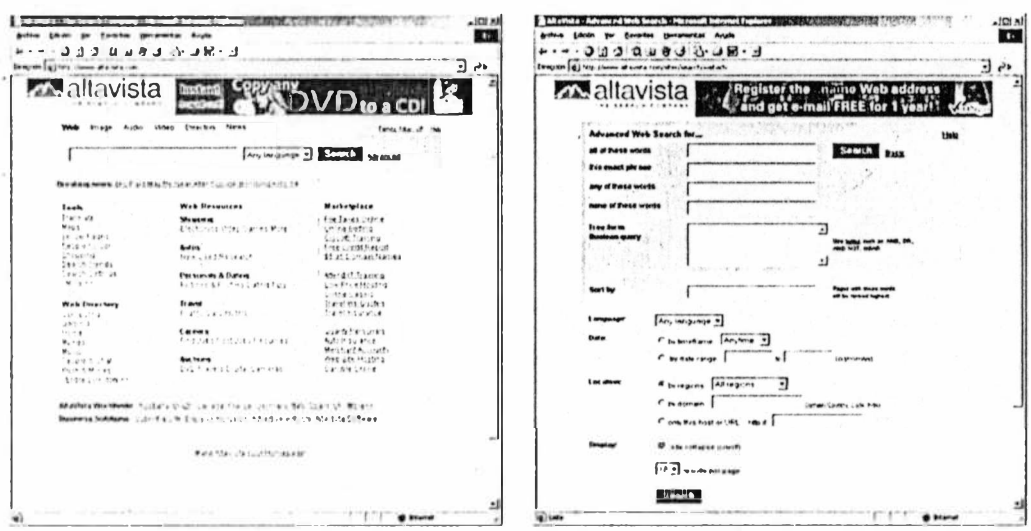


Figura 54. An example of *Simple Search Interfaz*: in www.altavista.com . Note that it is possible to access to a more sophisticated search facility if desired.

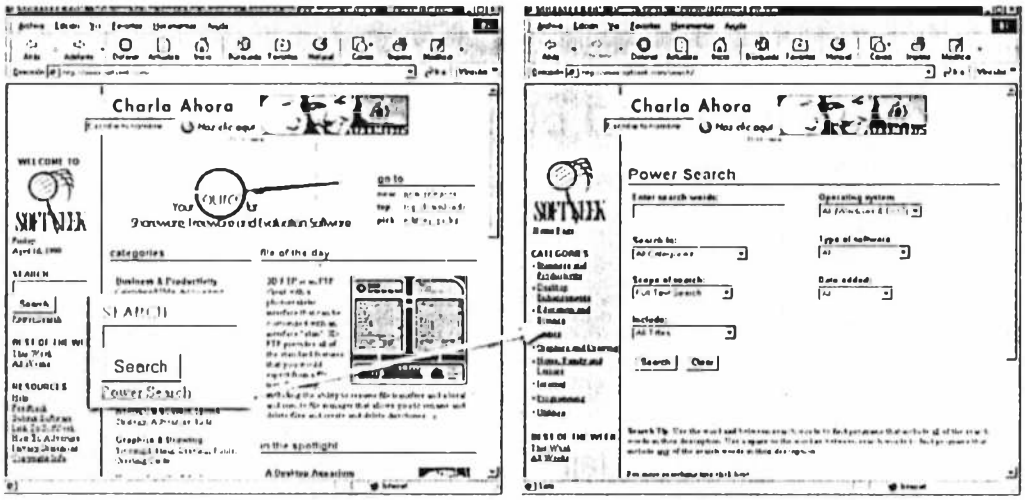


Figura 55. Otro ejemplo de Simple Search Interface en www.softseek.com. Una interfaz sencilla está siempre presente para el usuario, aunque también se encuentre disponible una máquina de búsqueda mas completa.

Patterns relacionados

Behavioral Grouping [Garrido97], ya que la habilidad para elegir opciones de búsqueda más complejas debe ser clara y visible en el contexto del campo de búsqueda.

12 Patrones de Comercio Electrónico

12.1 Introducción

12.2 Links Oportunistas (Opportunistic Linking)

Objetivo

Mantener atento al usuario interesado en el sitio. Seducirlo para navegar en el sitio incluso cuando él ha encontrado ya lo que él buscaba.

Motivación

Supongamos que estamos construyendo un comercio virtual tal como www.amazon.com. Entrando en el sitio podemos comprar muchos diversos productos tales como videos, libros o CDs. Podemos explorar los productos, y además de nosotros proporcionamos a links a las recomendaciones, a comentarios sobre los productos, a noticias, al etc. Sin embargo, muchos usuarios navegan con un objetivo específico: por ejemplo comprar un libro determinado. Una vez que hayan comprado ese libro, probablemente abandonen el sitio.

Una posibilidad es agregar links a cada página del producto para motivar al lector a navegar a otros productos. No obstante, en un sitio bien estructurado, debemos intentar proveer links con una la semántica fuerte para reducir el riesgo de la desorientación. Se presenta entonces el problema de cómo reconciliar estos dos requisitos.

Fuerzas

Deseamos mantener al usuario navegando en nuestro almacén incluso después él comprara algo.

No deseamos comprometer la estructura del almacén agregando links que no sean relevantes.

Solución

Mejorar la topología de links sugiriendo nuevos productos para explorar a partir de un producto dado. Utilice relaciones con una semántica fuerte para hacer la sensación del usuario confortable. Considere que muchos de estos links pueden cambiar cotidianamente, aspecto que deberá considerarse en el diseño de la interfaz. Es importante notar que este pattern se puede utilizar también en el nivel conceptual para derivar nuevas relaciones.

Sin embargo, el objetivo es claramente navegacional: mantener al usuario navegando de una manera agradable.

Ejemplos

Los "Links Oportunistas" se pueden encontrar en muchos sitios de comercio electrónico. Por ejemplo en www.cdnow.com o www.amazon.com se puede encontrar el CD relacionado con el que usted eligió.

En el cuadro 1 mostramos un ejemplo extremo en Amazon.com. Una vez que el usuario haya elegido un libro y lo haya puesto en la tarjeta de las compras, él recibe una sugerencia de otro libro que podría ser de su interés.

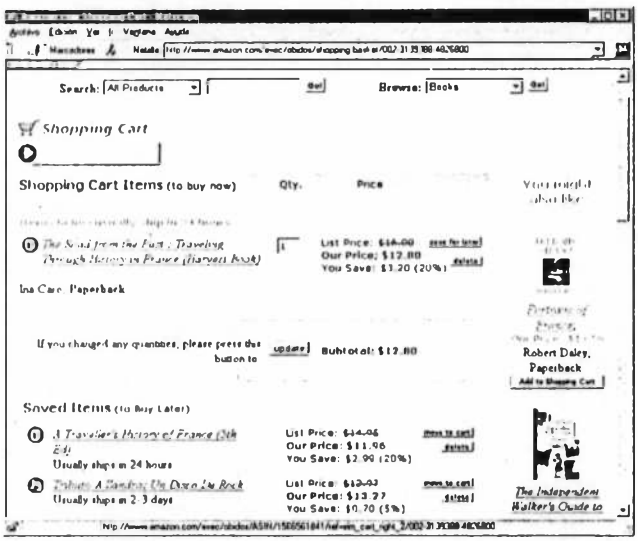


Figura 56. Ejemplo de Link Oportunista en www.amazon.com

Consecuencias.

Mantiene al usuario interesado en el sitio; lo cual acarrea como una consecuencia indirecta la mejora de las ventas.

Demasiados links (puede causar cognitive overhead).

Un modelo de datos subyacente más complejo.

La implementación de este modelo no es compleja pues no agrega nada nuevo a una aplicación convencional del Web. Solamente se requiere adaptar la interfaz mostrando más links (como en el cuadro 1). Esos links se pueden calcular directamente de los algoritmos que usan modelo de los datos subyacentes para encontrar el similaridades. Estos algoritmos pueden ser directos: por ejemplo encontrando algunos libros con la palabra clave simple que corresponde con (el mismo tema, e.g. recorrido) o más compleja según lo discutido en el pattern "Advising".

Patterns relacionados

Opportunistic linking es similar a "Advising"; de hecho, podría decirse que lo simplemente se esta realizando una mera aplicación del pattern. Sin embargo, el objetivo es diferente. Mientras que los "opportunistic links" intentan mantener al usuario dentro del sitio dándole nuevas ideas de comprar, mientras que "Advising" lo ayuda a elegir lo que él desea.

12.3 Consejo (Advising)

Objetivo

Ayudar al cliente a encontrar un producto en el almacén. Asistirlo según sus deseos.

Motivación

Habitualmente muchos usuarios entran en un comercio virtual apenas para encontrar algún producto quisieran comprar. En un almacén típico puede haber millares de productos y el proporcionar buenos índices a los motores de la búsqueda puede ser solo una solución parcial para darle una cierta orientación. Aplicando el patterns "News" [Rossi99] podemos mostrarle productos nuevos. Sin embargo, las noticias pueden cambiar de cotidianamente y además no podemos asegurar que el usuario estará interesado en un producto nuevo.

Fuerzas.

Los clientes en un comercio electrónico pueden necesitar ser asistidos para encontrar un producto.

Los motores y los índices de la búsqueda (por ejemplo taxonómicos) son útiles pero ellos pueden ser una solución incompleta.

Debemos considerar lo que puede desear el usuario.

Solución

Construir una funcionalidad especifica para aconsejar sobre productos. Estas funciones se pueden poner en ejecución de diversas maneras. Por ejemplo, puede haber un subsistema completo para las recomendaciones como en www.amazon.com que utilicen perfiles de los clientes (en general su historial de compras) para los productos de recomendación. También puede ser más general, y presentar al usuario los productos del mejor vendedor, o productos en liquidación, etc. El diseño de la funcionalidad de "consejo" no debe interferir con la estructura navegacional de la aplicación.

Ejemplos

"advising" se utiliza en casi todos los comercios virtuales. Por ejemplo, Amazon no solamente proporcionar a recomendaciones según el perfil de usuarios, también incluye a los mejores vendedores, poniendo al día cada hora su lista de 100 libros mas vendidos.

En www.barnesandnoble.com por ejemplo, una sección del negocio se incluye en el Home Page junto con recomendaciones generales. www.netgrocer.com solamente brinda información sobre productos en venta. En el cuadro XXX mostramos la estructura de aconsejar en www.cdnow.com. En www4.activebuyersguide.com uno puede encontrar un sitio completo dedicado a aconsejar. Muchos sitios de E-comerce proporcionan links a este sitio para proporcionar a descripciones de sus productos.

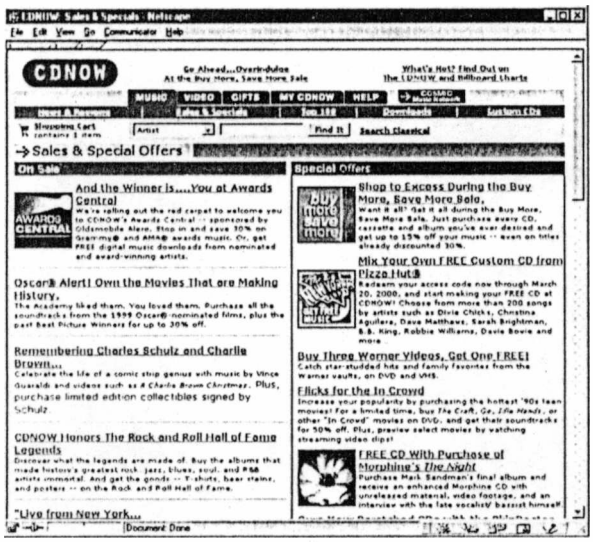


Figura 57. Advising in www.cdnow.com

Consecuencias

Ayudar a los clientes a decidir qué comprar.

La ayuda puede inducir a los clientes hacia ciertos productos.

Es necesario guardar el perfil del cliente y la historia.

Requiere algoritmos complejos

Implementación

La implementación pude realizarse de diversas maneras. La más simple debe incluir recomendaciones simples en la Home Page (como uno puede encontrar por ejemplo en Amazon.com). Esta implementacion necesita solamente esas recomendaciones (por ejemplo los titulos mas vendidos) para ser mostrados en la pagina. Una implementacion más sutil no pierde de vista el perfil de usuario (en la base de datos subyacente) registrando lo que él compró en el pasado y sugerir productos similares. La interfaz es similar a una pagina convencional.

Patrones relacionados

Advising es similar a *Opportunistic Linking*. Sin embargo, en este ejemplo el objetivo es ayudar al usuario a encontrar un producto, mientras que Opportunistic link trata de interesar al usuario una vez este encontró un producto. Advising tiene un alcance más general. De hecho, ambos modelos se podían considerar como versiones específicas del más genérico. El aconsejar se puede también combinar con los modelos de las noticias(NEWS) [Rossi99].

12.4 Proceso Explicito (Explicit Process)

Objetivo.

Ayudar al usuario a entender el proceso de compra cuando este no es atómico.

Motivación

El proceso de compra en la mayoria de los comercios electrónicos es complejo e involucra diferentes formularios (dirección del envío y de la factura, información sobre tarjeta de crédito, etc). Considerando la misma naturaleza del Web el usuario

puede experimentar la desorientación y puede preguntarse si él había llenado ya una cierta información y puede confundirse sobre el proceso entero.

Esto puede generar que cliente cancele el proceso o intente retroceder a las paginas anteriores, quizás arribando a un estado inconsistente (ver Backtrack seguro).

Fuerzas

El proceso de registración o compra puede ser complejo.

Los usuarios tienden para sentirse desorientados mientras avanzan en este de proceso

Solución

Dar al usuario un feedback perceptible sobre el proceso manteniéndolo actualizado sobre qué pasos de progresión él ya ha logrado. Esto puede ser realizado utilizando una línea del progreso. o apenas enumerando dichos pasos de progresión e informando a donde él ahora está. Considere las consecuencias posibles de su vuelta hacia atrás (ver SecureBacktrack) para reducir al mínimo las posibilidades de alcanzar estados inconsistentes. Los ejemplos en la mayoría de los almacenes podemos encontrar puestas en práctica simples de este modelo. Por ejemplo en www.barnesandnoble.com por ejemplo, el cliente tiene que progresar en 7 pasos de progresión que se indiquen claramente de una manera secuencial. En www.amazon.com mientras tanto la implementación es más elegante según lo mostrado en el cuadro 3. En la tapa de la pantalla usted puede ver una línea de proceso el indicar de que usted está en Items. Del paso de progresión y de que usted todavía falta cuatro pasos de progresión hasta acabar.

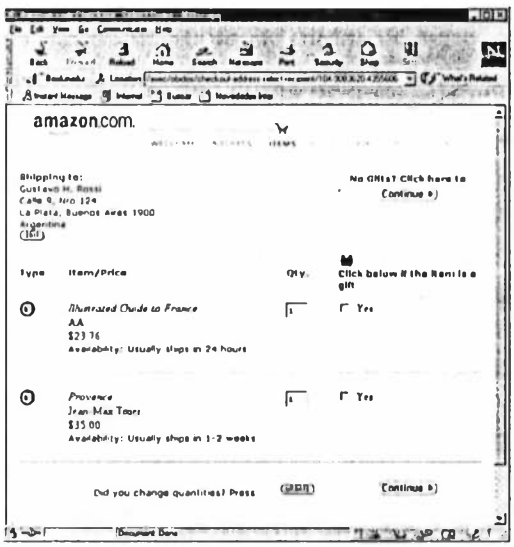


Figura 58. Ejemplo de Proceso Explícito en Amazon.com

Consecuencias.

Ayuda a entender procesos complejos. Ayuda puede para simplificar el proceso que deshace. Es necesario guardar el estado de user.s (en una sesión). Puede complicar el interfaz usuario La implementación la implementación de este modelo es bastante simple, puesto que no se ocupa de la lógica del negocio sino proporciona algo a un feedback visual al usuario del proceso actual. Aunque es posible realizar una cierta solución de la cara del cliente, la información que se mostrará es proporcionada

generalmente por la separación y es puesta en ejecución por el HTML llano o una combinación del HTML y de gráficos Modelos Relacionados: El proceso explícito se relaciona con seguro retrocede mientras que tienden para simplificar y para hacer el proceso de la comprobación (o el otro no-atómico) más seguro. El proceso explícito es un caso determinado del proceso FeedBack [Rossi00]

12.5 Desahacer Sencillo (Easy Undo)

Objetivo.

Proporcionar la capacidad de deshacer acciones realizadas e procesos complejos en forma segura.

Motivación

Como previamente fue explicado, un cierto proceso en aplicaciones de comercio electrónico puede ser muy complejo (como el proceso de checkout). Mientras que indicar el paso de progresión en el cual estamos es útil, puede también suceder que el usuario deletreó una cierta información mal o puede desear cambiar ciertos datos de su pedido. La solución simplista dejaría retroceder al cliente (con su botón BACK del browser), corregir los datos correspondientes y hacer de nuevo el proceso. Sin embargo, como el usuario puede no estar enterado de la semántica exacta de la navegación, puede suceder que se encuentre en un estado inconsistente. Por ejemplo, la cesta de compras pudo haber cambiado (si algunos ítems fueron cambiados por ejemplo). El problema con el botón "BACK" es que la semántica de retroceder puede interferir seriamente con esperada operación de UNDO.

Fuerzas.

La comprobación o el proceso de la colocación puede ser complejo.

El cliente puede necesitar deshacer una cierta operación anterior.

Usar el botón posterior puede ocasionar resultados inesperados.

Solución

Proveer al usuario con facilidades para deshacer una operación, desalentándole la utilización de los recursos de navegación para este propósito. Las facilidades para deshacer tendrán que considerar el estado del cliente en el de proceso para ser eficaces, y considerar la semántica de navegación para no alcanzar un estado inconsistente. Este patrón amplía la idea del backtracking típico en las aplicaciones del Web adaptándolo a la semántica de la aplicación. En vez de volver a la última página (usando el botón "back"), volvemos al estado correspondiente para deshacer la operación. Es importante remarcar que esta diferencia es específica a este dominio pues el Web se basa en un paradigma simple del hipertexto con una funcionalidad de backtracking general.

Ejemplos

Hemos encontrado muchos diversos ejemplos y puestas en práctica de este patrón en aplicaciones del comercio electrónico. Por ejemplo en www.powells.com, exponen a un cliente a toda la información en el paso de confirmación y él puede elegir actualizar o modificar sus datos. Mientras tanto en www.amazon.com, la información está agregada incrementalmente y el usuario puede elegir cambiar datos previamente incorporados.

Es importante notar los botones pequeños cerca de cada ítem de la información, indicando que usted puede corregir la información. Cuando se selecciona corregir la dirección de la factura por ejemplo, es posible volver a la página correspondiente, y una vez cambiado usted puede continuar el proceso naturalmente. Estamos utilizando un algoritmo backtrack-forward que cambie la semántica de navegación usual para adaptarla a las necesidades del almacén virtual.

Consecuencias.

Positivas:

Simplifica el deshacer operaciones.

Ayuda a solucionar la tensión entre backtracking y deshacer.

Negativas:

Requiere una interfaz elaborada.

Puede requerir almacenar el estado de cliente.

Implementación

Como ocurre con muchos aspectos de implementación en aplicaciones web, la estrategia adoptada para solucionar un problema tiene muchas variantes debido a la naturaleza conversacional inherente de la interacción en comercio electrónico. El registro del detalle de una actividad del cliente (la "memoria" de una conversación entre vendedor y cliente, o del "estado" de tal interacción) es un componente dominante para implementar la solución.

	Server-Based		Client-Based		OnTheWire	
Server-side	JSSP JSP / EJB ASP / COM CGI/ISAPI	EJB ASP-COM JSS Web Frameworks Database	JSSP JSP / EJB ASP / COM CGI/ISAPI	Cookies XML	JSSP JSP / EJB ASP / COM CGI/ISAPI	HiddenFields Fat URLs
Client-Side	Vbscript Javascript Java Applet	RDS Javascript	Vbscript Javascript Java Applet	Cookies XML	Vbscript Javascript Java Applet	HiddenFields Fat URLs

Tal estado se puede centralizar, "OnTheWire", "Client-Side" o "server-side", según la decisión de los desarrolladores sobre la localización determinada de la información. Una vez la información sobre la actividad del usuario se recupera (-independientemente de la estrategia utilizada -), la implementación real se reduce al problema de establecer la transición correspondiente al un instante determinado.

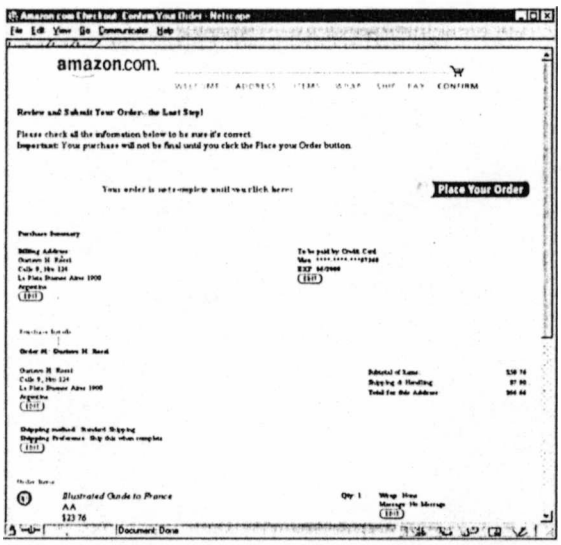


Figura 59. Ejemplo de Easy Undo en Amazon.

13 Patrones de Procesos

13.1 Introducción

Este conjunto de patrones describe procesos habituales presentes en las aplicaciones de todos los días. Cuando una de las responsabilidades de un sistema es actual como "guía" de un documento particular o del propio usuario para que éste logre completar una determinada tarea, se dice que el sistema implementa un "sistema de workflow". Un sistema con tales responsabilidades se dice que posee "requerimientos de workflow" [Meszaros97].

A continuación se incluyen 2 proto-patrones actualmente en estudio para analizar los procesos más habituales en aplicaciones de software.

13.2 Completar para Continuar (Complete to Proceed)

Intención

Establecer un conjunto de parámetros para realizar una tarea y un orden en el cual estos parámetros deben ser establecidos.

Problema

La facilidad de uso se menciona a menudo como un elemento esencial en muchos nuevos sistemas de software. Desafortunadamente, dada la creciente sofisticación de estos sistemas, continúan siendo difíciles de utilizar. Un problema recurrente es la disponibilidad cada vez más habitual de herramientas más inteligentes y automatizadas para eliminar tareas repetitivas y engorrosas. Claro que eso tiene un precio: esas herramientas deben ser parametrizadas. Finalmente, una restricción adicional es que dicha parametrización no puede ser realizada en cualquier orden, ya que habitualmente existen numerosas dependencias entre las opciones disponibles de acuerdo a lo que se establezca.

Vemos este problema recurrentemente, sean instaladores de software, herramientas de desarrollo, o el mismo proceso de pago (o "checkout") de un sitio de comercio electrónico en Internet. Todos ellos comparten la necesidad de establecer un gran número de parámetros, y parte de los mismos dependen de los valores establecidos en otros parámetros. Claramente existe la necesidad de establecer un orden de modo de poder llevar a cabo esta tarea.

Figura 60

Solución

Establecer las dependencias entre todos los parámetros, establecer un orden entre ellos y agruparlos en conjuntos donde no interfieran entre ellos. Luego, establecer las etapas sobre la base de estos conjuntos abstraídos, permitiendo retornar sobre aquellas etapas que hayan sido completadas e impidiendo acceder a las etapas posteriores mientras no se haya cumplido la regla de validación de la etapa actual.

Ejemplos

El ejemplo más común de este patrón son los "wizards", popularizados originalmente por Microsoft, en un esfuerzo por hacer las aplicaciones mas fáciles de usar. Otro ejemplo aparecen en los sitios de comercio electrónico durante el CheckOut.

Finalmente, un ejemplo en aplicaciones desktop aparece (además de los productos Microsoft) en herramientas de desarrollo como Eclipse o Builder, en sus herramientas de creación de un nuevo proyecto de software.

13.3 Point of No Return

Intención

El usuario no debe puede volver sobre sus pasos para mantener el estado consistente del proceso.

Problema

Existen actividades que pueden volverse hacia atrás, modificarse o incluso eliminarse sin que ello afecte la validez del resultado final del proceso, como por ejemplo los mencionados en 13.2.

Sin embargo, no todas las tareas son reversibles. Un ejemplo de este tipo de tareas son los pagos a través de medios electrónicos y, en general, todas aquellas de naturaleza transaccional, donde la única forma de revertir una acción es a través de otra cuyo efecto contrarreste la anterior pero no la hace desaparecer. Otro ejemplo son todas aquellas involucradas en un proceso productivo que involucre aplicación de materiales.

Solución

Dadas las características de la Web, una de las estrategias más comunes es establecer la expiración de la validez de las páginas anteriores.

Figura 61

Ejemplos

Este patrón habitualmente se encuentra en los procesos de pago, al momento de confirmar el pago con una tarjeta de pago. Una vez enviada la petición el proceso no puede ser cancelado ni es posible navegar hacia atrás.

Otro ejemplo se encuentra habitualmente en sistemas como el webmail, donde luego de seleccionar un mail para ser leído, no es posible ir hacia atrás sin recibir el mensaje de error "*su página ha caducado*". Esto es correcto, ya que el mensaje recién leído volvería a aparecer como "no leído" cuando eso claramente no es cierto.

14 Conclusiones

En este trabajo hemos abarcado los diferentes aspectos involucrados en la construcción de aplicaciones hipermediales, desde los estrictamente tecnológicos a los conceptuales. Hemos discutido arquitecturas para la implementación de aplicaciones Web, sus beneficios y propuesto mejoras.

El eje central de este trabajo ha sido la reusabilidad en la construcción de software, la abstracción de problemas de diseño recurrentes y la posibilidad de trasmitir la experiencia en el diseño, y en particular su aplicación a los sistemas de información hipermediales en la Web.

La reusabilidad en el software tiene varios aspectos, muchos de los cuales han sido experimentados en este trabajo a través del desarrollo y utilización de frameworks. Los frameworks abstraen el conocimiento de un dominio particular, las interacciones y el flujo de control entre las diversas componentes de software.

El desarrollo del framework OOHDM para aplicaciones Web ha sido un proceso largo y complejo, a lo largo de diversas iteraciones, y sometido a las tensiones de una tecnología subyacente de rápida evolución y redefinición. El resultado de este proceso es un framework cuyo aporte al desarrollo de las aplicaciones web es la provisión de mejores mecanismos de navegación, y mejores abstracciones conceptuales para un diseño que no dependa de una tecnología particular para ser utilizado.

El segundo aporte importante es la introducción del modelado de procesos como parte integral del diseño de una aplicación Web, lo cual simplifica el desarrollo de componentes reutilizables y amplia la flexibilidad de una aplicación para soportar cambios y extensiones en su funcionalidad. El modelado de procesos ataca el problema de sistematizar una de las áreas que habitualmente imponen cambios y disminuyen las posibilidades de reutilización de componentes. Como un aporte complementario a la introducción de la modelización de procesos, se proporciona un framework que implementa los aspectos conceptuales más importantes y permite aplicar estos conceptos a una aplicación real.

El tercer aporte, y quizás sea el de mayor relevancia, es la utilización de los patrones de diseño como un mecanismo que permita diseñar mejores aplicaciones. Hasta ahora, las metodologías de diseño de aplicaciones de software han provisto herramientas útiles para la expresión y modelización de aplicaciones pero, por su naturaleza, no proporcionan al diseñador novato herramientas para ayudar mejorar la calidad de su trabajo. Por primera vez, gracias a la aplicación de patrones de diseño, tenemos la posibilidad de trasmitir la experiencia y lecciones aprendidas en los diseños exitosos. Hemos compilado con importante número de patrones en un catálogo y creado además, un lenguaje que nos permite hablar con un mayor nivel de abstracción acerca de los diseños que realizamos. Gracias a esto, se ha abierto un campo de estudio que nos permitirá saber más acerca de porqué ciertas aplicaciones son mas efectivas y fáciles de usar.

Finalmente, durante el desarrollo de la herramienta gráfica para el modelado con OOHDM, se utilizó un framework existente (JHotDraw) para desarrollar gráficos técnicos, y se lo extendió para que soportara Java2D, una API estándar que provee la funcionalidad gráfica requerida para el desarrollo de la herramienta y soporte de las notaciones. Como resultado colateral de esta acción, el autor de este trabajo fue incorporado como miembro del equipo de desarrollo open source de dicho framework (https://sourceforge.net/project/memberlist.php?group_id=12679).

Durante el desarrollo del ambiente, se detectó la necesidad de simplificar el desarrollo de herramientas wizzard, para lo cual se desarrollo un micro-framework para simplificar el desarrollo a la especificación de la actividad a realizar, evitando la re-implementación de la mecanismo lógica de este tipo de ayudas.

14.1 Posibles Extensiones

-Ampliación de la utilización de reflexión para mejorar la integración del framework OOHDM con aplicaciones existentes.

-Estudio y ampliación del framework OOHDM para aplicaciones con necesidades de gran escalabilidad y basadas fuertemente en bases de datos relacionales.

-Ampliación del framework de procesos para que incluya patrones de uso.

-Ampliación del framework de procesos para que la definición de los mismos pueda ser transmitida en XML.

-Ampliación del framework de OOHDM para soportar mecanismos intercambiables de customización y personalización.

-Ampliación de la herramienta gráfica para ayudar a aplicar el framework de OOHDM.

-Implementación de mayor cantidad de patrones de diseño como constructos de modelización conceptual.

-Construcción de un wrapper para integrar el mini-framework para la construcción de wizzards con el modelo de procesos, para unificar .

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

15 Bibliografía

[Alexander77] **"A Pattern Language"** B. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I.Fiksdahl-King, and S. Angel, Oxford University Press, New York 1977.

[Apache] Apache's Java Apache Project page: http://java.apache.org

[Cocoon] Homepage del framework Cocoon/Apache: http://xml.apache.org/cocoon/index.HTML

[Enhydra] Documentación del framework Enhydra: http://xmlc.enhydra.org

[FreeMarker] Homepage del proyecto FreeMarker: http://freemarker.sourceforge.net

[Jakarta] Homepage Proyecto Jakarta: http://jakarta.apache.org

[JSP_Tomcat] Homepage del JSP engine: http://jakarta.apache.org/tomcat

[Jsp00] **"The Problems with JSP"** Jason Hunter (*Servlets.com*, January 2000):

http://www.servlets.com/soapbox/problems-jsp.HTML

[Jsp00b] "Reactions to 'The Problems with JSP" Jason Hunter (*Servlets.com*, February 2000):

http://www.servlets.com/soapbox/problems-jsp-reaction.HTML

[Lee et al.] **"A Pattern for Managing Distributed Workflows"** Seung Il Lee, Dongsoo Han, and Dongman Lee.

[Lyardet_b1] **"Capturing and Using Design Experience in Web Information Systems"**. F. Lyardet, G. Rossi and D. Schwabe. Handbook of Internet Computing, CRC Press, 2000.

[Lyardet00] **"Design & Architectural reuse in Comercio electrónico applications"**. Fernando Lyardet, Gustavo Rossi, Matias Butti, Dario Silva and Juan Danculovic. IeC2000 Conference November 2000, England.

[Lyardet01] **"Web Usability Patterns"**. Fernando Lyardet, Gustavo Rossi. EuroPLoP'00 Conference. Kloster-Irsee, Germany 2001

[Lyardet96a] **"Bridging the Gap Between the Hypermedia Design and Implementation"**, Poster. F. Lyardet, G. Rossi, Poster. ACM Hypertext'96.

[Lyardet96b] **"Enhancing Productivity in the Development of Hypermedia Applications"**, F. Lyardet, G. Rossi. Workshop of Next Generation Case Tools'96 (NGCT'96), held in CAiSE'96 Conference, Crete, Greece.

[Lyardet97] **"The Template Pattern"**, F. Lyardet, Proceedings of PLoP'97 Conference on Pattern Languages of Design, Sep/1997, Urbana-Champaign, Illinois.

[Lyardet98] **"Using Design Patterns in Educational Multimedia applications"**, F. Lyardet, G. Rossi, D. Schwabe. AACE ED-Media'98. Freiburg, Germany.

[Lyardet98b] **"Engineering Multimedia User Interfases with Objects and Patterns"**, F. Lyardet, G. Rossi, D. Schwabe. 2nd Workshop on Multimedia Software Engineering (MSE-IEEE), ICCSE'98 Conference. Kyoto, Japan.

[Lyardet98c] **"Discovering and Using Design Patterns in the WWW"**, Fernando D. Lyardet, Gustavo H. Rossi, Daniel Schwabe. Proceedings of the Brazilian Symposium on Multimedia and Hypermedia 1998. Rio de Janeiro. Brazil.

[Lyardet98d] **"Patterns for Designing Navigable Information Spaces"**, Gustavo H. Rossi, Daniel Schwabe, Fernando D. Lyardet. PLoP'98: Conference on Pattern Languages of Design, Sep/1998, Urbana Champaign, Illinois, USA.

[Lyardet99] **"Putting Design patterns to Work: a Case Study"**, Fernando D. Lyardet, Barbara Mercerat, Leonardo Miaton. 2nd Hypermedia Software Engineering Workshop. Hypertext'99, Darmstad. Germany.

[Lyardet99] **"User Interfaz Patterns for Hypermedia Applications"**, Fernando Lyardet, Gustavo Rossi and Daniel Schwabe. Submitted to INTERACT'99 HCI Patterns workshop. Edinburg, England 1999.

[Lyardet99] **"Patterns for Adding Search Capabilities to Web Information Systems"**, F. Lyardet, G. Rossi, D. Schwabe. EuroPLoP'99 Conference. Kloster-Irsee, Germany 1999.

[Lyardet99b] **"Patterns and Components: Capturing the Lasting amidst the Changing"** Hans-W. Gellersen, Fernando Lyardet, Martin Gaedke, Daniel Schwabe and Gustavo Rossi. The Active Web day, British Computer Society Special Interest Group in Human Computer Interaction, January 1999

[Meszaros97] **"A Pattern Language for Workflow Systems"**, G. Meszaros, K. Brown, in Proceedings of 4th Pattern Languages of Programs (PLoP) Conference, September 1997.

[MVC] Paper original sobre el MVC de Steve Burbeck:

http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.HTML

[Nielsen99] **"Website Usability"** Jacob Nielsen. New Riders Publishers, 1999.

[Patow99] **"The Recursion Pattern"**, G. Patow, F. Lyardet. To be published on EuroPLoP'99 Proceedings. Kloster-Irsee, Germany 1999

[Preece] Preece et al.: **"Human Computer Interaction"**. Addison Wesley, 1994.

[Reflection] Guia de reflectividad de Java (parte of the documentation that comes with the JDK: http://java.sun.com/j2se/1.3/docs/guide/reflection/index.HTML

[Rossi_b1] **"Patterns for Designing Navigable information Spaces"**. G. Rossi, D. Schwabe and F. Lyardet. "Pattern Languages of Program Design IV", Addison Wesley, 1999.

[Rossi_J1] ***"Integrating Patterns into the Hypermedia Development Process"***. G. Rossi, D. Schwabe and F. Lyardet. New Review of Hypermedia and Multimedia, Taylor Graham, 1999.

[Rossi_J2] **"Designing hypermedia applications with objects and patterns"**. G. Rossi, F. Lyardet and D. Schwabe. ACM Computing Surveys, 1999

[Rossi_J3] **"Improving Web Information Systems with navigational patterns"** D. Schwabe, G. Rossi, F. Lyardet. International Journal of Computer Networks and Applications, IEEE Press, 1999

[Rossi00] **"Patterns for Comercio electrónico Applications"**. Gustavo Rossi, Fernando Lyardet, Daniel Schwabe. EuroPLoP'00 Conference. Kloster-Irsee, Germany 2000

[Rossi00b] **"User Interfaz Patterns for Hypermedia Applications"**. Gustavo Rossi, Daniel Schwabe, Fernando Lyardet. Advanced Visual Interfases 2000. Palermo, Italy 2000

[Rossi99] **"Improving Web Information Systems with navigational patterns"**, Gustavo H. Rossi, Daniel Schwabe and Fernando Lyardet. Proceedings of WWW8 Conference. Toronto, Canada 1999

[Rossi99b] G. Rossi, D. Schwabe and F. Lyardet: **"Patterns for Designing Navigable Information Spaces"**. To be published in the book "Pattern Languages of Programs IV", Addison Wesley, 1999.

[Rossi99c] **"Web application models are views on conceptual models"** (G. Rossi, F. Lyardet, D. Schwabe). Workshop on the WWW and Conceptual Modeling, Paris, November 1999, LNCS Springer Verlag, forthcoming

[RossiHans01] **"Modelling and Designing Business Processes in Web Applications"**, Hans Albrecht Schmid, Gustavo Rossi, Fernando Lyardet

[Schwabe_b1] **"Web Design Frameworks: An Approach to Improve Reuse in Web Applications"** Daniel Schwabe, Gustavo Rossi, Luiselena Esmeraldo, and Fernando Lyardet. "Web Engineering: Managing Diversity and Complexity in Web Application Development", LNCS Vol. 2016, Springer-Verlag, April 2001.

[Schwabe_J4] ***"Engineering Web Aplications for Reuse"*** D. Schwabe, Luiselena Emeraldo, G. Rossi and F. Lyardet. IEEE Multimedia, 2001. Volume 8 Number 1.

[ServletsRes] Servlets resources: http://www.servlets.com

[Struts] Homepage del framework Struts: http://jakarta.apache.org/struts

[Sun_JSP] Homepage de JSP en Sun Corporation: http://www.javasoft.com/products/jsp

[Sun_servlets] Homepage de servlets de Sun Corporation: http://www.javasoft.com/products/servlet

[SunRef] **"Avoid scriptlets and expressions"**, http://java.sun.com/blueprints/guidelines/

[Turbine] Homepage del framework Turbine/Apache: http://java.apache.org/turbine

[Velocity] Homepage del proyecto "open source" Velocity: http://www.webmacro.org

[w3xsl] The World Wide Web Consortium's XSL page: http://www.w3.org/Style/XSL

[Kim90] **"Introduction to object-oriented databases."** KIM, Won. Ed. MIT Press, 1990. 234 p.

[Meszaros97] **"A Pattern Language for Workflow Systems"**, Gerard Meszaros, Kyle Brown. Proceedings of PLoP'97 Conference.

[w3] W3 Consortium. Http://w3.org/html/

[Woolf94] **"Improving Dependency Notification"**, Woolf, Bobby. The SmallTalk Report (a Sigs publication), Vol4 No3,1994.

[Woolf96] **"The Type Object Pattern"**, Woolf, Bobby. "The Type Object Pattern" . 1996.

Proceedings de Conferencias

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

ACM Hypertext 93, 94, 96, 97,98,99. ACM Publishing

ACM multimedia '97

ACM UIST '96, '95, '97

DONACION..... TES
$..... 02/4
Fecha..... 17-10-05
Inv. E. CA-187 Inv. B..... 2201