Modelando Interfaces para Aplicaciones Web

Luis A. Guerrero

Departamento de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
luis.guerrero@dcc.uchile.cl

Abstract. Muy poco de lo aprendido en el área de Ingeniería de Software se aplica en el desarrollo de sistemas Web, entre otras cosas, porque hay diferencias importantes en cuanto al diseño, desarrollo y mantenimiento de los sistemas Web, respecto de las aplicaciones tradicionales. Parte importante de un sistema Web tiene que ver con su contenido, su diseño gráfico y su modelo de navegación, además de su funcionalidad. En el presente artículo se muestran algunas taxonomías para sistemas Web, y se revisan algunas de las principales propuestas para modelar las interfaces de estos sistemas.

1. Introducción

El desarrollo y crecimiento de la Web han sido vertiginosos y sin precedentes durante los últimos años, en cuanto a número de usuarios conectados, cantidad de sitios o portales Web, y cantidad y tipo de herramientas para construir páginas y sitios Web. Este crecimiento ha provocado un significativo impacto en áreas tales como: negocios, finanzas, entretenimiento, comunicación, educación, gobierno, industria, e incluso en nuestra vida personal. Existen numerosos lenguajes y tecnologías relacionadas con la programación de aplicaciones que permiten generar páginas Web, no sólo del lado del servidor (*server-side Web pages*) sino también del lado del cliente (*client-side Web pages*). Esta enorme cantidad de recursos, y esta característica dual de las aplicaciones Web (que poseen un conjunto de funcionalidades independientes del lado del cliente y del lado del servidor), dificulta enormemente el análisis y diseño de este tipo de aplicaciones.

Las nuevas tendencias hacia el comercio electrónico, el trabajo en la casa, la transformación de aplicaciones tradicionales (*legacy systems*) a aplicaciones con interfaces Web, y la expansión de Internet hacia nuevos servicios y hacia otras áreas como la televisión, los teléfonos celulares (tecnología WAP), e incluso algunos electrodomésticos, hace pensar que este gran auge va a continuar por mucho más tiempo. Se van a requerir, entonces, aplicaciones cada vez más sofisticadas, pero con interfaces más claras y fáciles de usar, que puedan llegar a más gente. También es de esperar que se desarrollen más y mejores herramientas para el desarrollo de aplicaciones y páginas Web, que apoyen tanto su funcionalidad como el diseño gráfico de su interfaz y su contenido.

Podríamos decir que el actual desarrollo de sitios y aplicaciones Web está caracterizado por cuatro importantes factores: (1) las aplicaciones y sitios Web son cada vez más complejos en cuanto a gráfica, contenido y funcionalidad; (2) cada vez hay más y mejores herramientas de desarrollo; (3) los tiempos de desarrollo requeridos por las empresas son cada vez más cortos, para estar mejor posicionados que la competencia; (4) las aplicaciones y sitios Web requieren cambios periódicos tanto de contenido como de gráfica, para mantenerse atractivos a los usuarios, es decir, requieren un gran esfuerzo en mantenimiento.

Estos cuatro factores, entre otros, contribuyen a que el actual desarrollo de aplicaciones y sitios Web sea hecho *ad hoc* para cada proyecto, lo que requiere, en el corto plazo, un gran número de parches y cambios. Murugesan *et. al* sostienen que "*el desarrollo de sistemas basados en Web, carece de rigor, de un enfoque sistemático, de control de calidad y de aseguramiento de la calidad"* [1]. La alta probabilidad de fallas de los sistemas Web construidos hasta ahora (por falta de metodologías más rigurosas), y el hecho de que, al hacerse los sistemas cada vez más complejos, una falla puede ser propagada a muchos lugares a la vez (un error puede ser ejecutado muchas veces en muchos sitios simultáneamente), pueden provocar un quiebre irreparable de la confianza de los usuarios en el Web, causando así lo que ha sido definido como la *crisis del Web* [2].

Para evitar una posible crisis del Web, y lograr éxito en el desarrollo de aplicaciones Web cada vez más complejas, hay una imperiosa necesidad por enfoques disciplinados, y nuevos métodos y herramientas de desarrollo y evaluación de sistemas basados en Web [1]. Esta rigurosidad quizás debería partir por el diseño de la interfaz. La interfaz del sitio o aplicación Web es lo primero que piensa el cliente que solicita un proyecto de desarrollo sobre Web, y por tanto debería ser uno de los primeros artefactos que se diseñen, contrario al desarrollo de aplicaciones tradicionales, donde el diseño de la interfaz se realiza en una de las etapas finales del ciclo de desarrollo.

En el presente artículo se hace una revisión de las más recientes tecnologías para modelar y diseñar aplicaciones Web, principalmente aquellas que ponen énfasis en la forma de modelar interfaces. En la sección 2 se discuten algunas características de los sistemas Web, y se muestran algunas taxonomías. En la sección 3 se presentan algunas de las metodologías de desarrollo de proyectos Web más referenciadas en la literatura. La sección 4 presenta la propuesta de la arquitectura J2EE para el desarrollo de aplicaciones sobre Internet. Finalmente, en la sección 5 se discuten algunas diferencias entre sistemas tradicionales y sistemas Web, y se presentan algunas conclusiones sobre la importancia de utilizar métodos formales para el desarrollo de aplicaciones Web, principalmente para el diseño y construcción de sus interfaces.

2. Taxonomías Web

La poca rigurosidad en el desarrollo de sitios Web se nota desde la terminología utilizada. Términos como *aplicación Web* y *sitio Web* tienen distintos significados para distintos autores. Varios autores han definido taxonomías para tratar de clasificar

los distintos tipos de aplicaciones Web. Por ejemplo, Ginige y Murugesan [7] definen las siguientes categorías de aplicaciones Web: informativas (noticias en línea, servicios de noticias, catálogos, manuales), interactivas (formularios de registro, servicios en línea), transaccionales (compras electrónicas, bancos en línea), sistemas de workflow (planificación en línea, administración de inventarios, monitoreo), ambientes de trabajo colaborativo (sistemas distribuidos de autoría, herramientas colaborativas), comunidades en línea (grupos de chat, sistemas de recomendación), portales Web (centros comerciales electrónicos, intermediarios en línea).

Por su parte, Pressman define las siguientes categorías de aplicaciones Web [8]: informativas (se proporciona un contenido sólo de lectura con navegación y enlaces simples), de descarga (el usuario descarga información desde el servidor), personalizable (el usuario personaliza el contenido según sus propias necesidades), interactivas (comunicación entre comunidades de usuarios), con entradas de usuario (el ingreso de información por parte del usuario a través de formularios es el mecanismo primario), orientadas a transacciones (solicitudes del usuario al servidor Web), orientadas a servicios (proporcionan servicios a los usuarios), portales (el usuario navega en la aplicación y esta lo lleva a servicios fuera del dominio del portal), con acceso a bases de datos (el usuario extrae información de una base de datos), almacenes de datos (el usuario consulta información en una colección grande de datos).

Jim Conallen trata de establecer algunas diferencias entre los sistemas Web desde el punto de vista de la modificación de la lógica del negocio. Según Conallen, una *aplicación Web* es un *sistema Web* (servidor Web, red, protocolo HTTP, y browser) en el cual el usuario, a través de navegación y entrada de datos, afecta el estado del negocio [3]. De este modo, un *sitio Web* es un sistema Web donde la navegación del usuario no modifica lógica de negocio, o un sistema Web donde no hay lógica del negocio. Sin embargo, esta clasificación es un poco confusa. Una buena parte de los sistemas Web existentes extraen parte de la información que presentan a los usuarios desde bases de datos, y ocasionalmente modifican esta información, dependiendo de las acciones del usuario. De las cuatro operaciones básicas para el manejo de datos: crear (*insert*), recuperar (*select*), modificar (*update*) y borrar (*delete*), es común que los sistemas Web utilicen la segunda para desplegar información. Sin embargo, son las otras tres las que modifican el estado de una base de datos (el estado de la lógica del negocio), y es la presencia de estas operaciones, según la definición de Conallen, lo que define la funcionalidad de un sistema Web: en qué proporción la interacción del usuario con el sistema permite modificar el estado de los datos del sistema.

Por otra parte, la clasificación dada por Conallen no contempla la *complejidad* de los sistemas Web, la cual es de vital importancia en la Ingeniería Web para la construcción de las aplicaciones y para su futuro mantenimiento. Por ejemplo, según la clasificación de Conallen, algunos motores de búsqueda (*searh engines*) serían catalogados como “sitios Web”, ya que la navegación e interacción de los usuarios con el sistema no afecta el estado de la lógica del negocio, pues solamente son ejecutadas instrucciones de tipo *select* para obtener la información presentada. Y claramente un sitio Web de este tipo es mucho más complejo que un portal donde se muestre, por ejemplo, la cartelera cinematográfica de la semana. No obstante, estos dos sitios Web tienen algo en común, y es que la información que despliegan a sus usuarios cambia periódicamente, es decir, el contenido del sitio cambia.

Podríamos entonces decir que un sistema Web tiene tres componentes fundamentales: el diseño gráfico, el contenido y la funcionalidad. Es muy importante diferenciar estos aspectos pues el perfil de la persona encargada de implementar cada componente, es distinto. De acuerdo con estos tres nuevos criterios, podríamos definir una nueva taxonomía para sistemas Web. Los sistemas Web se clasificarían dependiendo de la cantidad y tipo de código (HTML, scripts y otros lenguajes de programación) que posean, según la cantidad de información que contengan, y según la calidad y complejidad del diseño. La figura 1 muestra esta taxonomía, compuesta por un sistema con tres dimensiones: contenido, diseño y funcionalidad [4].

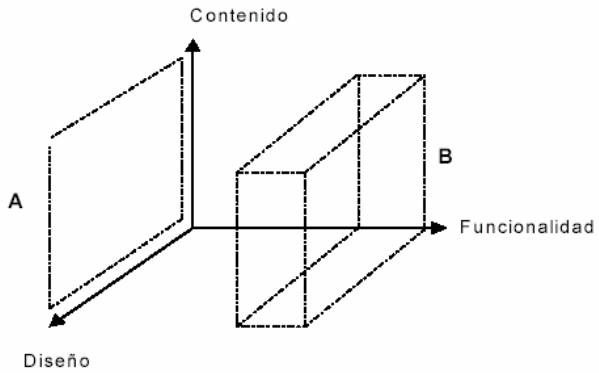

Fig. 1. Taxonomía de sistemas Web según su diseño, contenido y funcionalidad.

En la taxonomía de la figura 1, el plano de la zona *A* determina los sistemas que no contienen código ejecutable, es decir, los sistemas que no poseen funcionalidad. Estos sistemas podrían ser llamados *portales* o *sitios Web*. La zona *B*, por el contrario, define los sistemas que contienen mucha funcionalidad, independientemente del contenido y del diseño. Estos sistemas podrían llamarse *aplicaciones Web*. Pero, ¿qué pasa con un sistema Web que no cae en ninguna de las dos regiones señaladas?, ¿sería considerado un sitio o una aplicación Web? En realidad lo más importante es estimar el tiempo y esfuerzo en el desarrollo del sistema Web. El límite que hace la diferencia entre *sitio* y *aplicación*, dependerá de las métricas de estimación de cada empresa. Para nuestros efectos, diremos que una *aplicación Web* es aquella que requiere suficiente esfuerzo en cuanto a su funcionalidad, como para que se requiera un ingeniero de software.

A partir de la anterior taxonomía, es posible clasificar los sistemas Web de acuerdo con el tipo de código que poseen sus páginas, o al tipo de código que construye esas páginas. Esta taxonomía podría ser de utilidad para una empresa de desarrollo que necesite estimar el tiempo y el esfuerzo requerido para que sus distintos profesionales (ingenieros de software, diseñadores gráficos y arquitectos de información) desarrollen un proyecto Web.

3. Metodologías de desarrollo

En la literatura pueden encontrarse varias metodologías que sugieren diversas formas de enfrentar el desarrollo de una aplicación Web. Algunas de estas metodologías se revisan a continuación.

3.1. Iweb

Iweb (Ingeniería Web) [8] aplica un enfoque genérico de estrategias, tácticas y métodos especializados. Se deben aplicar las mismas tácticas de aseguramiento de la calidad aplicadas en todos los proyectos de ingeniería de software. Las aplicaciones basadas en Web incluyen sitios completos, funcionalidad especializada dentro de los sitios Web y aplicaciones de procesamiento de información que residen en Internet o en una intranet. Según los creadores de Iweb, algunos de los atributos que se encuentran en la mayoría de las aplicaciones Web son: intensivas en red, controladas por el contenido, evolución continua, inmediatez, seguridad, estética. Las actividades a realizar en proyectos de este tipo son: formulación del proyecto, planificación, análisis, ingeniería, generación de páginas y pruebas, y evaluación del cliente. Durante la formulación del proyecto se identifican las metas y objetivos de la aplicación Web y se establece el ámbito del primer incremento, se establece por qué es necesaria la aplicación, y quién la va a utilizar. Durante la planificación se estima el costo total del proyecto, se evalúan los riesgos asociados al esfuerzo de desarrollo, y se define un plan de trabajo. En el análisis se establecen los requisitos técnicos y los requisitos de diseño gráfico, se realiza un análisis de contenido, un análisis de la interacción, un análisis funcional, y un análisis de la configuración. Durante la ingeniería se diseña el contenido, y se realiza la producción. Paralelamente se diseña la arquitectura, la navegación y la interfaz. Durante la etapa de generación de páginas y pruebas, se revisa la navegación, se depuran los applets y otros scripts, y se prueba la aplicación en varios browsers. Finalmente, durante la evaluación del cliente se solicitan cambios, se integran de forma incremental, y se validan.

3.2. WSDM

WSDM (Web Site Design Modeling) [9] centra la generación del diseño en el usuario más que en los datos. Para esto trata de definir las “clases de usuarios” que visitarán el sitio. Según estas futuras visitas, y la forma en que estos usuarios recorrerán el sitio, se establecen los parámetros de diseño. El método está compuesto por cuatro fases: modelado de los usuarios, diseño conceptual, diseño de implementación, e implementación. En la primera fase se define el tipo de información que buscarán los usuarios cuando ingresen al sitio. En el diseño conceptual se modela la información requerida y se detallan las clases de usuarios. También se crea el diseño de navegación. En el diseño de la implementación se especifican los requisitos y restricciones del diseño gráfico del sitio, según lo definido en el diseño conceptual. Finalmente, en la implementación se selecciona el ambiente de desarrollo y se

implementa el sitio. WSDM se centra principalmente en el desarrollo de sitios Web de información (*kiosk Web sites*), más que en sitios interactivos o aplicaciones.

3.3. WebML

WebML (Web Modeling Language) [10] es un lenguaje conceptual para apoyar las actividades del diseño de sitios Web. Provee gráficos, formalismos, especificaciones, y diseño de procesos apoyados por herramientas gráficas. Define cinco tipos de modelos: estructura, derivación, composición, navegación y presentación. En el modelo de estructura se definen las entidades o contenedores de datos y sus relaciones. En el modelo de derivación se definen diferentes vistas y agrupaciones de los mismos datos. En el modelo de composición se especifican las páginas que componen el hipertexto, así como el contenido de éstas. El modelo de navegación especifica los vínculos (*links*) entre páginas y entre unidades de una misma página. Finalmente en el modelo de presentación se describe la apariencia gráfica de las páginas. WebML se enfoca en el diseño de la interfaz. Para esto provee una serie de estereotipos que pueden ser implementados usando XML.

3.4. WCPM

WCPM (WebComposition Process Model) [11] permite modelar componentes con distinto grado de funcionalidad, que van desde un simple link, hasta un script con cierta funcionalidad. Usando estos componentes es posible desarrollar una aplicación Web. Las aplicaciones así creadas son más fáciles de modificar y mantener. Este método, basado en la orientación a objetos, se enfoca en reutilizar el código y en mantener la integridad de las aplicaciones Web desarrolladas. Introduce conceptos de un modelo de procesos abierto, que permite desarrollar componentes reutilizables. Estos componentes se modelan usando el WebComposition Markup Language (WCML), que es una extensión de XML.

3.5. Web Engineering

Web Engineering [12] está compuesto por un conjunto de actividades necesarias para llevar a cabo el desarrollo de un sitio Web. Las actividades sugeridas por sus creadores son: análisis de contexto, modelo de productos, modelo de procesos, plan del proyecto, desarrollo, y mantenimiento. Durante el análisis de contexto se estudia el dominio del problema, identificando las posibles soluciones. Se analiza también el presupuesto y el tiempo de desarrollo. Usando un modelo de productos, las páginas Web pueden ser creadas por demanda, haciendo las consultas respectivas a la base de datos. El modelo de procesos considera las fases de desarrollo y la integración. Este modelo describe el tipo de desarrollo a seguir, según el proyecto. El plan del proyecto considera aspectos de gestión de proyectos, documentación, aseguramiento y control de calidad.

3.6. Otros métodos

RMM (Relationship Management Methodology) [13] se enfoca en el diseño y construcción de aplicaciones multimediales que tienen estructuras estables, pero con información altamente cambiante.

En JESSICA [14] se propone una metodología que cubre el ciclo de vida completo de una aplicación Web, basada en un modelo de abstracción orientado a objetos. Propone un lenguaje para modelar basado en XML, y un mecanismo para el mapeo automático del diseño a los recursos Web.

OOHDM (Object Oriented Hypermedia Design Method) [15] es un método basado en modelos para el desarrollo de sitios Web, sistemas de información, presentaciones multimediales, quiscos de información, etc. Se trata por separado el diseño de componentes, el modelo de navegación y el diseño de la interfaz.

Web Design Guidelines [16] es una guía planteada por IBM para el diseño de procesos centrados en el usuario. Contiene cuatro fases: planificación, diseño, producción y mantenimiento.

4. Arquitectura J2EE

En los últimos años, el proceso de desarrollo de software empresarial ha sufrido varios cambios, y en el centro de estos cambios se encuentra, sin lugar a dudas, la plataforma Java 2 Edición Empresarial o J2EE (Java 2 Enterprise Edition). J2EE provee una plataforma unificada para el desarrollo de aplicaciones distribuidas basadas en servidores.

La arquitectura J2EE es multicapa, y está formada por cuatro capas: capa cliente, capa Web, capa de negocios, y capa de sistemas empresariales de información o sistemas legados [5]. Las relaciones entre los componentes de las distintas capas se muestra en la figura 2.

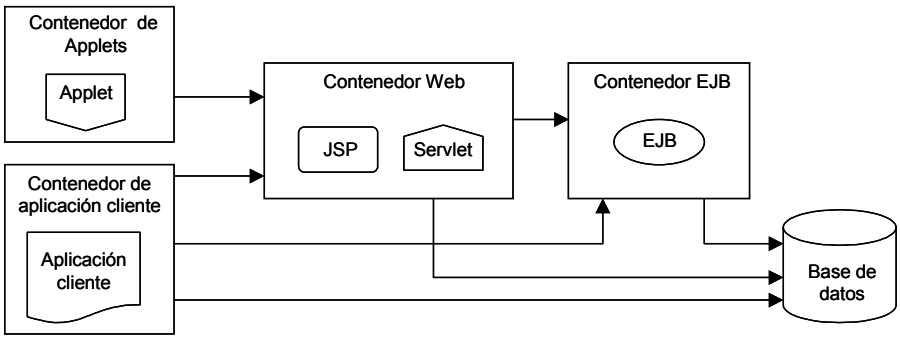

Fig. 2. Arquitectura J2EE.

La *capa del cliente* interactúa con el usuario y le muestra a éste la información que le envía el sistema. Algunos ejemplos de clientes son: clientes HTML, applets Java y

aplicaciones Java. La *capa Web* genera la lógica de la presentación (clientes HTML, applets Java y otros clientes Web) y acepta las respuestas que el usuario envía. Esta capa es implementada a través de servlets y JSP (Java Server Pages). La capa de negocio administra la lógica del negocio de la aplicación. Los componentes de negocio típicamente se implementan como componentes EJB (Enterprise Java Beans). Finalmente, la capa de sistemas empresariales de información (EIS) incluye sistemas de bases de datos, sistemas transaccionales, y sistemas de planificación. Esta capa es el punto de enlace de la plataforma J2EE con sistemas empresariales no-J2EE, o sistemas legados.

Una muy buena y antigua práctica de programación sugiere separar la *interfaz* de la *lógica de la aplicación*. Este concepto está inmerso en la plataforma J2EE. Una *aplicación Web* es vista como una aplicación de software *normal*, pero con una interfaz Web. Por tanto, para modelar y diseñar una aplicación Web, se puede utilizar cualquier metodología de desarrollo de software (por ejemplo RUP o XP). La única diferencia radica en la etapa en que se modela la interfaz gráfica de usuario [6].

5. Discusión

Desde el punto de vista del desarrollo de proyectos, existen claras diferencias entre una aplicación Web y una aplicación tradicional. Donald Reifer [17] establece estas diferencias a través de las características mostradas en la tabla 1.

Las diferencias entre estos tipos de proyectos son de vital importancia en la selección de los perfiles del equipo de trabajo, y en la estimación del tiempo de desarrollo y del costo. Es muy difícil, o casi imposible, utilizar las mismas métricas en ambos tipos de proyectos.

Un enfoque que parece razonable, desde el punto de vista de las nuevas tendencias de desarrollo como la plataforma J2EE, parece ser la separación de un proyecto Web en dos sub-proyectos: uno referido a la funcionalidad de la aplicación, y otro referido al diseño gráfico y al contenido. El primer sub-proyecto se puede atacar utilizando la experiencia y metodologías del desarrollo tradicional de aplicaciones. El segundo sub-proyecto, enfocado en la interfaz de la aplicación (diseño gráfico y contenido) debe ser realizado utilizando paradigmas y metodologías no tradicionales en la Ingeniería de Software. El tipo de personal, así como la estimación de costos y tiempos varía para cada sub-proyecto.

La taxonomía presentada en la figura 1, podría entonces dividirse en dos partes: un plano bi-dimensional que muestre el contenido y diseño necesarios para la aplicación Web, y un eje que refleje la cantidad y complejidad de la funcionalidad requerida.

Los proyectos Web sin funcionalidad, o con poca funcionalidad, podrían desarrollarse utilizando alguna de las metodologías revisadas, pero para aplicaciones con cierto grado de funcionalidad quizás requieran una combinación de alguna de las metodologías revisadas, junto con las metodologías tradicionales para el desarrollo de aplicaciones.

Table 1. Proyectos de desarrollo Web versus proyectos tradicionales.

Característica	Desarrollo tradicional	Desarrollo Web
Objetivo primario	Productos de calidad al mínimo costo	Productos de calidad al mercado lo más rápido posible
Tamaño típico del proyecto	Mediano a grande (equipos de cientos de miembros)	Pequeños (equipos de 3 a 5 miembros)
Tiempo de desarrollo	10 – 18 meses	3 – 6 meses
Enfoque de desarrollo	Clásico, basado en requisitos, entregas incrementales, casos de uso, documentación	Desarrollo rápido de aplicaciones (RAD), agrupar bloques de construcción, prototipos, RUP
Tecnologías de ingeniería usadas	Orientación a objetos, lenguajes modernos, herramientas CASE, etc.	Métodos basados en componentes, lenguajes de cuarta y quinta generación, visualización, etc.
Procesos	Basados en CMM	Ad hoc
Desarrollo de productos	Sistemas basados en código, reuso, muchas interfaces externas, algunas aplicaciones complejas	Sistemas basados en objetos, componentes reutilizables, pocas interfaces externas, aplicaciones relativamente simples
Personal involucrado	Ingenieros de software profesionales con 5 o más años de experiencia en al menos dos dominios de aplicación	Diseñadores gráficos, ingenieros con poca experiencia (2 o más años), recién graduados
Tecnologías de estimación	Uso de datos históricos, modelos basados en puntos por función, WBS para proyectos pequeños	Uso de la actual experiencia, diseño ajustable basado en recursos disponibles, WBS para proyectos pequeños

Referencias

[1] Murugesan, S., Deshpande, Y., Hansen S., and Ginige, A. "Web Engineering: A New Discipline for Development of Web-based Systems". Proceedings of the International Conference on Software Engineering, ICSE'99, May, 1999, Los Angeles, U.S.A.

[2] Zelnick, N. "Nifty Technology and Nonconformance: The Web in Crisis". IEEE Computer, October, 1998, pp. 115-116 and 119.

[3] Conallen, J. "Modeling Web Application Architectures with UML". Communications of the ACM, 42(10), October, 1999, pp. 63-70.

[4] Bravo, C., y Guerrero, L. A. "Métricas de Funcionalidad: Una Taxonomía para Sistemas Web". Actas del Primer Workshop de Ingeniería de Software, Punta Arenas, Chile, Noviembre, 2001.

[5] Alur, D., Crupi, J., and Malks, D. *Core J2EE Patterns*. Sun Microsystems Press, Prentice Hall, Palo Alto, California, U.S.A., 2001.

[6] Larman, G. *Applying UML and Patterns. An introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall, Second Edition, 2002.

[7] Ginige, A., and Murugesan, S. "Web Engineering: An Introduction". IEEE Multimedia, 8(1), pp. 14-18, 2001.

[8] Pressman, R. E. Ingeniería de Software, McGraw Hill, Quinta Edición en Español, 2002.

[9] De Troyer, O. “WSDM: A User Centered Design Meted for Web Sites”. In 7th World Wide Web Conference, Brisbane, Australia, 1998.
[10] WebML User Guide. http://www.WebML.org.
[11] Gaedke, G. “Development and Evolution of Web Application using the Web Composition Process Model”. International Workshop on Web Engineering at the 9th World Wide Web Conference, Ámsterdam, The Netherlands, May, 2000.
[12] Ginige, A., and Murugesan, S. “Web Engineering: An Introduction”. IEEE Multimedia, 8(1), pp. 14-18, 2001.
[13] Isakowitz, T. “RMM: A Methodology for Structured Hypermedia Design”. Communications of the ACM, 38(8), pp. 26-30, 1995.
[14] Barta, R., and Schanz, M. “JESSICA: An Object Oriented Hypermedia Publishing Processor”. In 7th World Wide Web Conference, Brisbane, Australia, 1998.
[15] Schwabe, D., Rossi, G., and Barbosa, J. “Systematic Hypermedia Design with OOHDM” ACM International Conference on Hypertext’ 96. Washington, USA, 1996.
[16] IBM Web Design Guidelines. http://www-3.ibm.com/ibm/easy/eou_ext.nfs/publish/558.
[17] Reifer, D. “Web Development: Estimating Quick-to-Market Software”. IEEE Software, November/December 2000, pp. 57-64.