MONOGRAFÍA

INGENIERÍA WEB

AISI - IWEB

INGENIERÍA WEB

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS Y TABLAS

INGENIERÍA WEB

1. Introducción

Las aplicaciones basadas en Web cada vez son más requeridas y demandadas a causa de la gran gama de funcionalidad que ofrecen a una gran cantidad de usuarios (Murugesan & Ginige, 2005, p.1). La confianza depositada en la información existente en la Web crece a un ritmo asombroso, prueba de ello es que desde marzo de 2005 hasta marzo de 2006 la cantidad de usuarios en el mundo aumentó de 888 millones a 1,023 millones lo que significa un incremento del 15,2% de usuarios en un año (Internet World Stats, 2006).

Pressman (2002, p.521) expone esta realidad al decir que "la World Wide Web (WWW) e Internet han introducido a la población en general en el mundo de la informática", en el sentido en que muchas actividades se realizan a través de la red, por ejemplo reservaciones de habitaciones en hoteles, reservas de pasajes de avión, venta de bienes y servicios, operaciones bancarias, la comunicación por medio de correo electrónico, el Chat y otros. Se puede decir, entonces, que Internet y la Web son los avances más importantes en la historia de la Informática.

Tal fenómeno ha despertado el interés de muchas empresas, negocios y organizaciones por estar presentes en la Web, para así ser más competitivas y reconocidas alrededor del mundo (Murugesan, Deshpande, Hansen & Ginige, 1999). El resultado de esta carrera por captar la atención del público en general es el desarrollo de aplicaciones cada vez más complejas y desafiantes. Desafortunadamente, dicha complejidad no parece estar acompañada de los mecanismos adecuados que garanticen la calidad de estos sistemas de los que cada día se tiene mayor dependencia a nivel social, funcional y económico.

Es lógico que esta situación genere la preocupación de la comunidad científica y técnica involucrada en el desarrollo Web. La búsqueda por desarrollar aplicaciones Web que tengan un buen desempeño, que sean confiables y de calidad se ha convertido en una tarea delicada, ya que la aplicación de las metodologías utilizadas para el desarrollo del software tradicional no ha sido suficiente, por que la Web posee características que agregan complejidad a las tareas de análisis, diseño, mantenimiento, etc. Situación irónica tomando en cuenta la cantidad de herramientas que permiten desarrollar aplicaciones Web incluso sin necesidad de poseer basto conocimiento de programación. A esto se suman las malas prácticas del desarrollo Web similares a las que se observan en la elaboración de software tradicional.

Pressman (1998, 2002) moderó una mesa redonda virtual con representantes de la ingeniería del software tradicional y del software basado exclusivamente en Internet. El debate principalmente se centró en discutir si valía la pena aplicar un proceso de ingeniería de las aplicaciones con base en Internet, o qué características tenían estas que justificaran el no utilizarlo. La conclusión general fue que aplicar un proceso de

ingeniería nunca es una mala idea, pero que éste debería adaptarse a los requerimientos de cambio continuo y rapidez siempre presentes en el proceso de desarrollo Web. A partir de esta iniciativa en los últimos años se han organizado congresos, talleres, debates, foros, simposios, etc., los que generalmente aportan material de investigación tratando de normar la producción de sitios Web.

Varios autores están de acuerdo en que si se quiere tener éxito al construir software basado en la Web se necesita adoptar un proceso de desarrollo disciplinado que sea eficiente y repetible, además de una sólida metodología (Ginige & Murugesan, 2001; Murugesan et al., 1999; Murugesan & Ginige, 2005; Olsina, 1999; Pressman, 2002). Los esfuerzos y resultados satisfactorios con respecto al tratamiento de esta tarea es lo que se viene a llamar Ingeniería Web *(IWeb)*. "Esta disciplina emergente de la Ingeniería Web se apoya en un enfoque integral y disciplinado orientado a un desarrollo exitoso de la Web" (Murugesan & Ginige, 2005, p.1).

Si bien la Ingeniería Web es relativamente nueva, va cobrando fuerza a medida que los investigadores reconocen su importancia. El camino por recorrer es todavía largo y arduo, pero es alentador si se considera que el material escrito, resultado de las nuevas investigaciones, está en continuo aumento y que la cantidad de autores fascinados por este tema crece también. La temática es bastante amplia, y en esta compilación introductoria se resumen los aspectos más relevantes para conocer los aspectos pasados, presentes y futuros de la Ingeniería Web.

2. Historia

La Ingeniería Web emerge progresivamente como una nueva disciplina que dirige las necesidades únicas del desarrollo de sistemas basados en Web y trata de normar y estandarizar la producción de estas aplicaciones; pero es desde 1998 que se empieza a nombrar en estos términos cuando se realiza el primer Workshop de Ingeniería Web en Brisbane, Australia de manera conjunta con la Conferencia WWW7, *World Wide Web Conference* (Deshpande, Ginige, Murugesan & Hansen, 2002; Murugesan et al., 1999; Murugesan & Ginige, 2005, p. 10;).

En ese mismo año, 1998, como mencionan Murugesan y sus colegas (1999), la revista *IEEE Software* organiza una interesante discusión en mesa redonda sobre la cuestión "¿Pueden las Aplicaciones Basadas en Internet ser tratadas con Ingeniería?". Como resultado se establece la necesidad de dirigir el estado del desarrollo de los sistemas basados en Web teniendo como base a la Ingeniería del Software, puesto que las tareas de programación y de desarrollo de software están involucradas, pero sin dejar de lado el análisis de las características de la Web (Pressman, 1998).

Desde entonces una buena cantidad de eventos similares se efectúan exclusivamente para tratar el tema de la Ingeniería Web tales como las conferencias WWW 1999-

2005, HICS[1] 1999-2001, SEKE[2] 2002 y 2003, entre otros; y la realizada anualmente Conferencia Internacional de Ingeniería Web, *International Conference on Web Engineering (ICWE)* 2002-2005 (Murugesan & Ginige, 2005, p. 10).

3. Panorama General de la Ingeniería Web

3.1. Evolución de la Web

La Web se ha integrado al ritmo de vida y trabajo de las personas en pocos años. Desde su inicial propósito de compartir información entre algunos pocos científicos, utilizando sitios Web simples que consistían primariamente en documentos de texto hiperenlazados, la Web ha crecido rápidamente en su alcance y extensión de uso, apoyándose en los avances tecnológicos de Internet y la Web además de los estándares. En 10 años el número de sitios Web ha crecido dramáticamente de 100 alrededor de 45 millones (Figura 1).

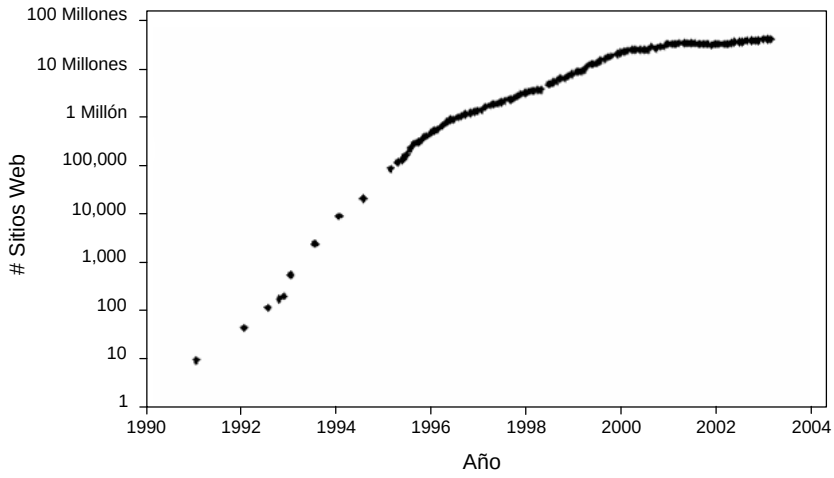

Figura 1: Crecimiento de los Sitios Web.
Fuente: (Murugesan & Ginine, 2005, p.4). Procedido de Hobbes' Internet Timeline, 2004, www.zakon.org/robert/internet/timeline/.

Nota: Sitios Web = Número de servidores Web; un host puede tener múltiples sitios usando diferentes dominios o números de puertos.

Las empresas, agencias de viaje, bancos, industrias, institutos educacionales, negocios de entretenimiento y gobiernos utilizan sistemas basados en Web de gran escala y aplicaciones para progresar, aumentar y/o extender sus operaciones. El comercio electrónico se ha vuelto global y general. Sistemas y bases de datos tradicionales están siendo progresivamente migrados a la Web. Las aplicaciones Web modernas se ejecutan en hardware distribuido y en sistemas de computadoras heterogéneos. Más aún, el mercado se ha llenado de tecnología avanzada como

[1] HICS, International Conference on High-Performance Computing
[2] SEKE, Software Engineering & Knowledge Engineering

computadoras portátiles y dispositivos de comunicación, y una nueva ola de aplicaciones Web móviles está emergiendo rápidamente. La Web ha cambiado la forma de vivir y trabajar en todo nivel (Murugesan & Ginine, 2005, p.4).

3.2. Crisis en la Web

El desarrollo Web es importante e indispensable, en muchos casos, para el trabajo, educación, comercio, entretenimiento, y muchas otras aplicaciones; es decir que se depende mucho del éxito de esta tarea. Sin embargo, grandes cantidades de información se manejan de una manera ad hoc[3] en el desarrollo y mantenimiento Web, resultando en sistemas y aplicaciones Web con una deficiencia en cuanto a rigor, técnicas sistematizadas, metodologías sólidas y aseguramiento de la calidad (Ginige & Murugesan, 2001; Murugesan & Ginige, 2005, p.2; Murugesan et al., 1999; Olsina, 1999).

Problemas tales como información irrelevante, dificultad en el uso, lentitud en la respuesta, roturas en la navegación y arquitectura, y posibles quebrantamientos a la seguridad del sistema, son comunes. Estos problemas son encontrados porque se falla al perfilar las necesidades del usuario, y debido a cuestiones como administración de contenido, mantenimiento, ejecución, seguridad y escalabilidad. A veces también se pasan por alto importantes consideraciones no técnicas como la propiedad y la privacidad (Murugesan & Ginige, 2005, p.2).

Los actuales problemas también son resultado del continuo avance en las tecnologías para la Web e Internet, el aumento de aplicaciones Web comerciales, la frenética demanda por estar en la Web, y la necesidad de migrar sistemas heredados[4] a ambientes de sistemas Web (Ginige & Murugesan, 2001).

Muchos desarrolladores Web parecieran pensar que elaborar una aplicación Web es simplemente crear páginas Web usando HTML o software tal como *Front Page* o *Dreamweaver* y plasmar unas cuantas imágenes y documentos hiperenlazados. A pesar de que en muchos casos una página Web simple es suficiente, en muchos otros casos las aplicaciones deben ser más complejas y necesitan cumplir con los requerimientos para que puedan evolucionar y cambiar. Hay más aspectos que considerar en el desarrollo Web qué sólo el diseño visual y la interfaz de usuario. También implica planificación de la arquitectura, diseño, pruebas, aseguramiento de la calidad y la evaluación de la ejecución, además de las continuas actualizaciones y el mantenimiento del sistema a medida que los requerimientos de uso crezcan y se desarrollen (Murugesan & Ginige, 2005, p.2).

---

[3] Ad hoc es una expresión adverbial latina que se aplica a lo que se dice o se hace sólo para un fin determinado (Prats & Rovira, 1992). Término aplicado a los métodos improvisados y no repetibles.

[4] Los sistemas heredados son aplicaciones cruciales para el éxito de negocios, que han servido por varios años y que están afectados por una enorme necesidad de ser reconstruidos o rehechos en su totalidad (Pressman, 2002, pp.542-557). Los sistemas heredados, en términos de desarrollo Web, se consideran a aquellos sistemas de escritorio que necesitan ser llevados a entornos de Red, para ser transformados en sistemas basados en Web.

Por lo tanto, el desarrollo ad hoc no es apropiado para grandes y complejos sistemas Web. Para construir sistemas y aplicaciones basadas en Web complejas y de larga escala con éxito se necesita adoptar un proceso disciplinado y una metodología sólida, utilizar herramientas más adecuadas, y seguir una colección de buenas guías (Murugesan & Ginige, 2005, p.2), es decir la Ingeniería Web.

3.2.1. Problemas en el Desarrollo Web

En el desarrollo Web, además, se lidia con ciertos problemas a la hora de poner en marcha un proyecto, puesto que los desarrolladores y administradores de las aplicaciones suelen adoptar malas prácticas, ya sea por la falta de métodos para el desarrollo Web, o por el desconocimiento de métodos o por la resistencia a utilizarlos. Ginige & Murugesan (2001) plantean a continuación situaciones problemáticas en un diagnóstico realizado sobre el desarrollo de sistemas basados en Web:

Los desarrolladores Web ponen poca atención a los requerimientos y al análisis, metodologías y procesos de desarrollo, la calidad, la evaluación del desempeño, a la administración de la configuración y del proyecto, a la mantenibilidad y escalabilidad. Además, que los desarrolladores confían en sus conocimientos y experiencia más que en prácticas estandarizadas. Estos sistemas también suelen tener escasa y deficiente documentación y sus técnicas de prueba son inapropiadas.

Los desarrolladores Web consideran el desarrollo Web, primordialmente, una cuestión de autoría más que una cuestión de desarrollo de aplicaciones. También se maneja el mito de que "el desarrollo Web es un arte" que debe tratar con "manipulación y presentación de medios". De seguro que el proceso de diseño y construcción tiene un importante aspecto artístico, sin embargo, los desarrolladores necesitan seguir alguna disciplina, como en la ciencia y la ingeniería.

A varios atributos de calidad en los sistemas basados en Web; tales como la navegación sencilla, la accesibilidad, la escalabilidad, la mantenibilidad, la usabilidad, la compatibilidad e interoperabilidad, y la seguridad y confianza; no se les da la consideración que merecen durante el desarrollo. Algunas aplicaciones Web también fallan en dirigir aspectos culturales, de privacidad, morales y legales.

El desarrollo de sistemas basados en Web no es un evento de una sola vez, como algunos practican, es un proceso con un largo ciclo de vida.

3.3. Definición de Ingeniería Web

La *Ingeniería Web* (IWeb) es una disciplina derivada de la Ingeniería del Software, compuesta de métodos y técnicas para desarrollar y mantener aplicaciones Web de calidad. Su existencia, comparada con otros tipos de ingeniería, es bastante reciente. Su estudio es tema de mucho interés en los últimos años. A pesar de que varios de sus métodos y técnicas están basados en los que componen a la ingeniería del

software, en muchos casos su tratamiento es diferente ya que las aplicaciones basadas en Web son un tipo de software con características especiales. La investigación en el área demuestra que la IWeb tiene el potencial para convertirse en una disciplina completamente diferente cuyo éxito en su aplicación puede ser fundamental para los negocios y para muchas otras áreas aplicables a la Web. Algunas definiciones hechas por autores destacados en el área se mencionan a continuación.

Definición 1

"La IWeb es la aplicación de principios específicos, ingenieriles y de administración, además de aproximaciones sistemáticas, para el desarrollo y uso exitoso, y para el mantenimiento de sistemas y aplicaciones Web de alta calidad." (Murugesan et al., 1999)

Definición 2

"La ingeniería Web esta relacionada con el establecimiento y utilización de principios científicos, de ingeniería y de gestión, y con enfoques sistemáticos y disciplinados del éxito del desarrollo, empleo y mantenimiento de sistemas y aplicaciones basadas en Web de alta calidad (Murugesan, 1999, citado por Pressman, 2002, p.522). Con el objeto de evitar una Web enmarañada[5] y lograr un mayor éxito en el desarrollo y aplicación de sistemas basados en Web. Tales enfoques y técnicas deberán tener en cuenta las características especiales en el medio nuevo, en los entonos y escenarios operativos, y en la multiplicidad de perfiles de usuario implicando todo ello un reto adicional para el desarrollo de aplicaciones basadas en Web." (Pressman, 2002, p.522)

Definición 3

"La ingeniería Web es una forma de elaborar y organizar conocimiento acerca del desarrollo de aplicaciones Web, y la aplicación de ese conocimiento, o el direccionamiento de nuevos requerimientos o retos. Esto impulsa un desarrollo de sistemas basados en Web bajo control, minimizando el riesgo y aumentando la calidad, mantenibilidad, y escalabilidad de las aplicaciones." (Murugesan & Ginige, 2005).

La definición 1 expone de manera concreta los objetivos de la IWeb al utilizar enfoques científicos para lograr el éxito en el desarrollo y utilización de aplicaciones Web. La definición 2 también expone medios y objetivos similares para el tratamiento de la Web, además establece que deben considerarse siempre las características de la Web ya que es un área que aporta retos nuevos para la ingeniería. Finalmente, la definición 3 sintetiza las definiciones anteriores de manera que deja en claro cuales son los fines del área, pero no deja en claro cuales

[5] "Web enmarañada. Esta frase connota un cúmulo de aplicaciones basadas en Web pobremente desarrolladas y con una probabilidad de fallo bastante alta." (Pressman, 2002, p.522).

son los medios para lograr esta tarea. En conclusión se observa que es la definición 2 la que manifiesta lo que este escrito necesita para la mejor comprensión del tema.

3.4. Categorías de las Aplicaciones Web

Las aplicaciones Web varían enormemente según el propósito de su funcionamiento: de aplicaciones de pequeña escala y corta vida (meses, semanas, incluso días) a aplicaciones empresariales de larga escala distribuidas por Internet, así como intranets y extranets de corporaciones (Murugesan & Ginine, 2005, p.5). No todas las aplicaciones y sistemas basados en Web comparten las mismas características y requerimientos (Deshpande, Chandrarathna, Ginige, 2002).

Las aplicaciones Web ahora ofrecen vasta variedad de funcionalidad y tienen diferentes características y requerimientos. Estas aplicaciones pueden ser categorizadas en muchas formas –no existe una única y aceptada forma. La categorización de las aplicaciones basadas en Web fundadas en la funcionalidad (Tabla 1) es útil para entender sus requerimientos y para desarrollar y desplegar sistemas basados en Web y aplicaciones (Murugesan & Ginine, 2005, p.5).

Las categorías representan muchos propósitos, requieren diferentes tecnologías y también determinan la posibilidad de evolucionar. Entonces, los sitios Web pueden extenderse a grades sitios dinámicos (Deshpande & Chandrarathna et al., 2002).

Tabla 1: Categorías de las Aplicaciones Web
Fuente: (Deshpande &, Chandrarathna et al., 2002; Ginige & Murugesan 2001; Murugesan & Ginige, 2005)

Categoría	Ejemplos
Informativas	Periódicos online, catálogos, boletines informativos, clasificados, manuales de servicio, libros electrónicos.
Interactivas Información provista por el usuario Acceso Personalizado	Formularios de registro, presentaciones informativas personalizadas.
Transaccionales	Tiendas electrónicas (virtuales), bienes y servicios online, bancos vía Web.
Orientada al flujo de trabajo (Workflow).	Sistemas de planificación y programación de actividades, administración de inventarios, monitoreo de estado.
Ambientes de trabajo de colaboración.	Sistemas de autoría distribuidos, herramientas de diseño colaborativas.
Comunidades Online, mercados.	Subastas, grupos de Chat, sistemas recomendados.
Portales Web.	Tiendas electrónicas, intermediarios.

3.5. Complejidad de las aplicaciones Web

Así como las aplicaciones Web han evolucionado; la demanda por los sistemas basados en Web y la complejidad en su diseño, desarrollo, mantenimiento y administración de estos sistemas ha crecido también. El diseño de estas aplicaciones demandan el balance entre la información del contenido, la estética y la ejecución (Ginige & Murugesan, 2001).

El avance en la tecnología de las herramientas de desarrollo Web también influye en el grado de complejidad de las aplicaciones, puesto que es posible construir desde páginas Web de propósito muy simple hasta publicar sistemas de software bastante complejos y sofisticados orientados a la Web. Esta diferenciación se la hace más clara definiéndose a las aplicaciones simples como estáticas y a las aplicaciones complejas como dinámicas. También vale la pena hacer una diferenciación en cuanto a su orientación, es decir, las aplicaciones orientadas a los documentos y las aplicaciones orientadas a las aplicaciones. En Olsina (1999, p.9) se presenta esta diferenciación gráficamente en la Figura 2 extraída a su vez de las categorías formuladas por Powell et al. (1998) las cuales se muestran a continuación:

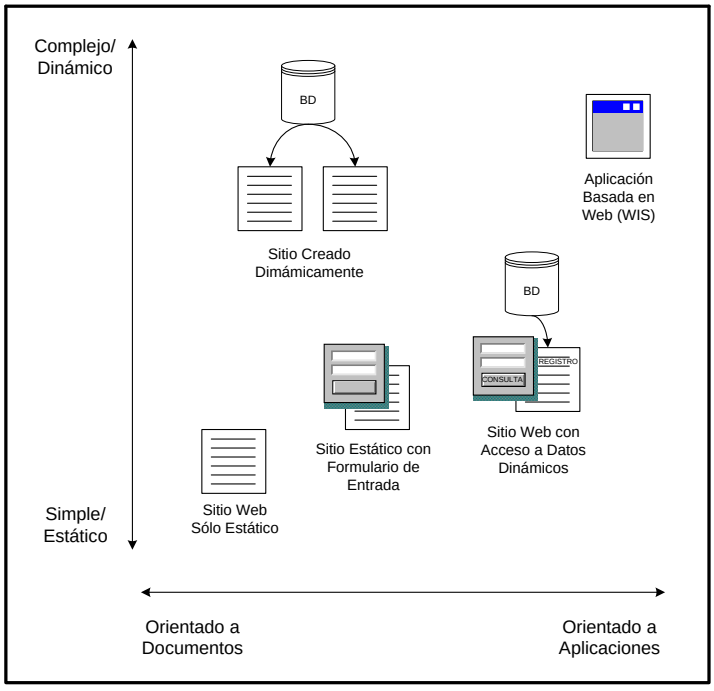

Figura 2: Rasgos de Complejidad y niveles de orientación de las Aplicaciones Web
Fuente: (Powell et al 98) modificado de Olsina (1999, p.10.)

Sitio sólo estático: Es una colección de páginas estáticas, es decir, documentos editados y publicados en HTML.

Sitio Estático con Formularios de Entrada: Permite manejar mecanismos de retroalimentación tales como cuestionarios, libros de invitados, o comentarios y sugerencias, son de fácil implementación favoreciendo a la comunicación en línea.

Sitio con Acceso de Datos Dinámicos: En estas aplicaciones el usuario puede acceder a las páginas (del lado del cliente) a datos almacenados en bases de datos remotas por medio de consultas y búsquedas.

Sitio creado dinámicamente: Este tipo sitios Web provee requerimientos personalizados en consideración con el contenido de las páginas para cada instancia de usuario; y permite construir dinámicamente sitios en conformidad con el entorno del navegador del usuario. Para ello, los documentos estáticos deben ser mudados dinámicamente aunque en el lado del cliente, no provean interactividad alguna.

Aplicación de Software basada en la Web: Este tipo de sitio Web (a veces implementado como Intranet o Extranet) puede ser un sistema de control y seguimiento de inventarios, o un sistema de educación a distancia, etc., proveyendo funcionalidad que está más cercana a una implementación cliente/servidor tradicional que a un sitio Web estático.

En Ginige &Murugesan (2001) se presenta, de forma comparativa, en la Tabla 2 otras características relacionadas con la complejidad de los sistemas Web, es decir, sistemas basados en Web simples y sistemas basados en Web avanzados.

Tabla 2. Características de sistemas basados en Web simples y avanzados. Fuente: (Ginige & Murugesan, 2001)

Sistemas Basados en Web Simples	Sistemas Basados en Web Avanzados
Simples páginas Web en principio, presentando información textual.	Páginas Web complejas.
El contenido de la información no cambia –bastante estático.	La información es dinámica –cambia con el tiempo y las necesidades del usuario.
Simple navegación	Difícil navegación y búsqueda de información.
Sistemas Autónomos.	Integrados con bases de datos y otros sistemas planificados, calendarizados y monitoreados.
Gran rendimiento no significa mayor requerimiento.	Requiere gran rendimiento y continua disponibilidad.
Desarrollados por un solo individuo o por un equipo pequeño.	Requieren un gran equipo de desarrolladores con experiencia en diversas áreas.
Usados para la diseminación de información.	Desplegados en aplicaciones de misión-crítica.

3.6. Herramientas y Tecnologías del Desarrollo Web

Una buena cantidad de diferentes herramientas y tecnologías existen que pueden ser usadas para el desarrollo Web. Los desarrolladores tienen que escoger las herramientas apropiadas, instalarlas y configurarlas de manera que el proceso de desarrollo sea productivo. Estas herramientas están construidas para dar soporte a una o más actividades, de hecho una herramienta puede incorporar una o más tecnologías (Shklar & Rosen, 2003, p. 337). La elección de la herramienta correcta depende de factores como la complejidad de la aplicación, a qué categoría pertenece y la arquitectura de desarrollo que se va a aplicar.

Este conjunto de variadas herramientas puede agruparse en cinco categorías básicas, presentadas a continuación (Fraternali, 1999):

Primeramente, los *editores visuales de HTML y administradores de sitios,* cuyos ambientes de administración de contenido originalmente fueron concebidas para aliviar la compleja tarea de escribir código HTML y de mantener las páginas de un sitio Web en los archivos de sistema. En una configuración típica estos productos constituyen editores HTML de tipo "lo que ves es lo que obtienes", *What You See Is What You Get (WYSIWYG)*, que permite al usuario diseñar páginas sofisticadas en HTML sin tener que programar, además de un administrador visual, que despliega gráficamente el contenido de la página y soporta funciones como la carga, borrado y renombrado de archivos, y con la detección y reparación de enlaces rotos

Las *herramientas de hipermedia* comparten el mismo enfoque de los editores visuales de HTML pero tienen un origen diferente, publicaciones hipermedia fuera de línea. Estos productos entraron al mercado del desarrollo de bases de datos para la Web y el interés sobre ellos está motivado por sus aproximaciones no convencionales al diseño de aplicaciones y su enfoque específico por la navegación y presentación. A pesar de que su arquitectura es inmadura, las herramientas hipermedia permiten diseñar sofisticadas interfaces de usuario, que exhiben un grado de control sobre precisión gráfica.

Los *integradores HTML-DBPL*, explícitamente dirigen la combinación entre aplicaciones Web y bases de datos y son muy poderosas. Son capaces de producir productos que pueden ser usados para implementar aplicaciones por encima de grandes bases de información, por otro lado su uso requiere típicamente un esfuerzo de desarrollo sustancial. Las soluciones existentes proponen diferentes medios de integración de la aplicación con bases de datos.

Los *editores de formas* Web, *escritores de reportes* y *los asistentes de publicación de bases de datos* aproximan bastante a la integración bases de datos a aplicaciones Web, dirigiendo la migración del cliente al servidor. Estas herramientas permiten aumentar la productividad de los desarrolladores en tareas tales como la edición de formas, la escritura de reportes, y la programación basada en eventos. Ofrecen un nivel más alto de soporte con respecto a los integradores HTML-DBPL.

Finalmente, *los generadores de modelos Web* son aquello productos que provén una cobertura completa de todas las actividades de desarrollo. Se tratan de herramientas que integran en una sola las capacidades de programación, conexión a base de datos e interfaces de usuario con diseño gráfico. Sin embargo algunos entendidos en desarrollo Web recomiendan en lo posible siempre hacer que las tareas de desarrollo sean independientes. Las ventajas de realizar esta separación se citan a continuación (Galli, 2001, Froufe, 2002):

> Primera y más importante, evita hacer dependiente todo un sitio Web (scripts, diseño gráfico y maquetación) de una sola tecnología o aplicación.
>
> El diseñador no tendrá que preocuparse de aspectos de programación o base de datos.
>
> Se puede usar cualquier software para el desarrollo del diseño de la página Web.
>
> Se evita a los programadores que aprendan a usar un determinado software de diseño o maquetación, que normalmente están orientados a diseñadores y con una curva de aprendizaje bastante elevada.
>
> Si se permite que el trabajo de los programadores sea lo más independiente del diseño posible, y además se le deja usar sus propias herramientas, se aumenta considerablemente la productividad.
>
> Existe más libertad para poder ser reemplazar o actualizar las herramientas de desarrollo, ya que estas evolucionan muy rápidamente. Lo que es bueno hoy, podría ser muy malo mañana.
>
> El involucrar a más niveles, es decir, más capas de desarrollo se consigue más mantenibilidad y escalabilidad en las aplicaciones, ya que se pueden incrementar los recursos más fácilmente si el tráfico aumenta. Además es posible modificar una parte del sistema, la base de datos por ejemplo, sin que otras áreas se vean afectadas.

Otro tipo de herramienta que capta la atención de los desarrolladores por su gran utilidad son la herramientas CASE[6]. Este término es generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar las tareas de desarrollo. Compiladores, editores de estructura, sistemas de control de recursos de código, y herramientas de modelado, son todas herramientas CASE. En los últimos años el término CASE ha sido asociado con familias específicas de herramientas que ayudan a definir y validar las especificaciones de un sistema o automatizar el proceso de construcción de software (Spector, 2002).

[6] Sigla conformada por Computer-Aided Software Engineering, que traducido al español vendría a ser Ingeniería de Software Asistida por Computadora

Finalmente, otro factor determinante que es necesario considerar cuando se tratan tecnologías y herramientas de desarrollo es la utilización de software libre o software propietario, ambos tipos de herramientas tienen sus ventajas y desventajas.

El *software libre* es aquel que puede ser distribuido, modificado, copiado y usado; por lo tanto, debe venir acompañado del código fuente para hacer efectivas las libertades que lo caracterizan. También es conveniente no confundir el software libre con el software gratuito, este no cuesta nada, hecho que no lo convierte en software libre, porque no es una cuestión de precio, sino de libertad (Culebro, Gómez & Torres, 2006).

El software no libre también es llamado *software propietario* se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o distribuirlo (con o sin modificaciones), o que su código fuente no está disponible o el acceso a este se encuentra restringido. En el software no libre una persona física o jurídica posee los derechos de autor sobre un software negando o no otorgando, al mismo tiempo, los derechos de usar el programa con cualquier propósito; de estudiar cómo funciona el programa y adaptarlo a las propias necesidades; de distribuir copias; o de mejorar el programa y hacer públicas las mejoras. De esta manera, un software sigue siendo no libre aún si el código fuente es hecho público, cuando se mantiene la reserva de derechos sobre el uso, modificación o distribución (Culebro et al., 2006).

El *software semilibre* es aquel que mantiene las mismas características que el software libre para los usuarios individuales, entidades educativas o sin ánimo de lucro, sin embargo prohíbe esas libertades para su uso comercial o empresarial (Culebro et al., 2006).

Tanto el software libre como el software propietario poseen ventajas y desventajas. Algunas de estas ventajas y desventajas pueden ser consideradas de formas diferentes por usuarios particulares, por las empresas, y las administraciones públicas (Culebro et al., 2006).

Las ventajas del software libre son las siguientes:

Bajo costo de adquisición y libre uso.
Innovación tecnológica más frecuente.
Requisitos de hardware menores y durabilidad de las soluciones.
Escrutinio publico, es decir que cualquier observador es libre de emitir opiniones y sugerir cambios.
Independencia del proveedor.
Este tipo de software es más adaptable a entornos cambiantes.
Existe mayos libertad para trabajar en diferentes lenguas incluso las lenguas minoritarias.

También se observan desventajas del software libre:

La curva de aprendizaje es mayor.
El software libre no tiene garantía proveniente del autor.
Se necesita dedicar recursos a la reparación de errores. Sin embargo en el software propietario es imposible reparar errores, hay que esperar a que saquen a la venta otra versión.
Las interfaces gráficas de usuario y la multimedia apenas se están estabilizando.
La mayoría de la configuración de hardware no es intuitiva.
El usuario debe tener nociones de programación.
La diversidad de distribuciones, métodos de empaquetamiento, licencias de uso, herramientas con un mismo fin, etc., pueden crear confusión en cierto número de personas.

El software propietario cuenta con las ventajas siguientes:

Las compañías productoras de software propietario por lo general tienen departamentos de control de calidad.
Además poseen recursos a la investigación y personal altamente capacitado.
Uso común por los usuarios, ya que es relativamente fácil encontrar a alguien que lo sepa usar.
Este tipo de software es para aplicaciones muy específicas
Amplio campo de expansión de uso en universidades
Difusión de publicaciones acerca del uso y aplicación del software.

Entre las desventajas del software propietario se pueden citar:

Cursos de aprendizaje costosos.
El funcionamiento del software propietario es un secreto que guarda celosamente la compañía que lo produce.
Es ilegal extender una pieza de software propietario para adaptarla a las necesidades particulares de un problema específico.
Las compañías tienen el derecho exclusivo de innovación.
Si la compañía fabricante del software propietario se va a la banca rota el soporte técnico desaparece, la posibilidad de en un futuro tener versiones mejoradas de dicho software desaparece y la posibilidad de corregir los errores también desaparece.
Si una compañía fabricante de software es comprada por otra más poderosa, es probable que esa línea de software quede descontinuada y nunca más en la vida vuelva a tener una modificación.

El software libre ha tenido un papel fundamental en el crecimiento y extensión de la Web, porque la mayor parte de la infraestructura de Internet se basa en protocolos abiertos. Aproximadamente el 67% de servidores Web emplean Apache, otro gran

número usan SendMail para gestionar el envío de correo electrónico y prácticamente la totalidad de los servidores de nombres (DNS). Es indiscutible la importancia que ha tenido el software libre en la extensión y el desarrollo de la Web desde sus inicios. Sin la existencia del software libre la Web hoy en día probablemente no existiría. Ha sido igualmente importante el hecho de que los protocolos que definen la arquitectura de Internet sean abiertos y que no hayan sido controlados por una o varias empresas (Culebro et al., 2006).

4. Desarrollo de Aplicaciones Web

4.1. Características del Desarrollo Web

Es importante entender que las aplicaciones Web tienen ciertas características que las hacen diferente frente al software tradicional, sistemas de información, o el desarrollo de aplicaciones de computadora. Las aplicaciones Web tienen las siguientes características: (Murugesan & Ginine, 2005; citando a Deshpande & Ginige et al. (2002); Deshpande & Hansen (2001); Ginige & Murugesan (2001a) y (2001b); Glass (2001); Lowe (2003), Murugesan et al., (1999); Pressman (2001) (2002) y (2004))

Las aplicaciones Web constantemente evolucionan. En muchos casos, no es posible especificar por completo lo que un sitio Web debe contener al iniciarse su proceso de desarrollo, debido a que su estructura y funcionalidad evolucionan con el tiempo, especialmente después de que el sistema es puesto en funcionamiento. A diferencia del software convencional que cumple una revisión planificada y discreta en tiempos específicos de su ciclo de vida, las aplicaciones Web constantemente evolucionan en términos de sus requerimientos y funcionalidad. Administrar estos cambios es un desafío técnico, organizacional y administrativo considerable -más demandante que el desarrollo de software tradicional.

Las aplicaciones Web son inherentemente diferentes del software. El contenido, que probablemente incluya texto, gráficos, imágenes, audio, y video, es integrado con proceso. También, la manera en que el contenido es presentado y organizado tiene implicaciones en la ejecución y tiempo de respuesta del sistema.

Las aplicaciones Web están hechas para ser usadas por una vasta y variada comunidad de usuarios, con requerimientos, expectativas y destrezas variadas. Entonces la interfaz de usuario y características de usabilidad tienen que conocer las necesidades de una diversa comunidad de usuarios a los que no se les puede ofrecer sesiones de entrenamiento, tampoco interacciones humano (Web, *Human-Web Interaction (HWI)*, interfaces de usuario y presentaciones de la información).

Buena parte de los sistemas basados en Web tienen el contenido administrado automáticamente (base de datos automatizada). El desarrollo de estos sistemas incluye la creación y administración del contenido, así como la apropiada previsión para la subsecuente creación, mantenimiento y administración después del desarrollo inicial.

En general, *muchos sistemas basados en Web demandan un buen "look and feel*[7]*,"* favoreciendo la creatividad e incorporación de multimedia en presentación e interfase. En estos sistemas, se pone más énfasis en la creatividad visual y la presentación.

Las aplicaciones Web tienen un cronograma de desarrollo comprimido, y la presión por el tiempo es grande. Por lo tanto, el desarrollo del proceso que pueda duran varios meses a un año o más no es apropiado.

Las aplicaciones Web son desarrolladas por un equipo de personas -normalmente jóvenes- considerado pequeño comparado con los equipos de desarrolladores de software, con diferentes habilidades, conocimientos y procedencias. Su percepción de la Web y de la calidad de los sistemas basados en Web también difiere ampliamente, causando confusión y resultando en malas guías de las prioridades.

Existe un cambio rápido y constante avance de las tecnologías para la Web, y estándares que ofrecen sus propios desafíos. Nuevos lenguajes, estándares y herramientas con los que hay que lidiar; además de varios errores en recientes versiones de los nuevos lenguajes, las herramientas de desarrollo, y los ambientes (inestabilidad tecnológica).

El desarrollo Web utiliza diversas tecnologías y estándares e integra numerosas variedades de componentes, incluyendo software tradicional y no tradicional. Además de intérpretes de lenguajes encriptados, archivos HTML, bases de datos, imagines, y otros componentes multimedia tales como video y audio, y complejas interfases de usuario.

Los medios de entrega para las aplicaciones Web son algo diferentes de los de software tradicional. Las aplicaciones Web necesitan arreglárselas con una variedad de dispositivos y formatos, y soportar hardware, software y redes con accesos a velocidades variadas.

Las necesidades de seguridad y privacidad son más demandantes en sistemas basados en Web que en los sistemas de software tradicional.

La Web ejemplifica un fuerte lazo entre el arte y la ciencia que generalmente tropieza con el desarrollo de software.

Estas características únicas de la Web y sus aplicaciones hacen del desarrollo Web una disciplina diferente y más desafiante que el desarrollo de software tradicional.

[7] Look and feel, traducido textualmente significa "mirar y sentir", se refiere a la sensación de satisfacción que se espera del usuario considerando el aspecto visual de la aplicación.

4.2. Modelos de Proceso de Desarrollo Web

Para desarrollar una aplicación Web es necesario primero entenderla. Para lograrlo, se deben definir los objetivos generales para su creación, sus requerimientos funcionales e informativos, los usuarios a los que va dirigida, y si se diera el caso de elaborar una aplicación dinámica o un sistema orientado a la Web, sus bases de datos y procedimientos. Estas actividades son la base de la IWeb. Los modelos se crean para entender mejor la entidad que se va a construir, en este caso una aplicación Web, y deben ser capaces de modelar la información y sus funciones (Pressman, 2002).

Las características de sistemas y aplicaciones basados en Web influyen enormemente en el proceso de la IWeb, dado que estas aplicaciones suelen ser controladas por el contenido haciendo hincapié en la estética, es probable que las actividades de desarrollo paralelas se planifiquen dentro del proceso IWeb y se necesiten un equipo de personas tanto técnicas como no técnicas. Elegir un adecuado modelo de procesos depende también de los requerimientos de la aplicación. Pressman (2002, p.525) enumera una serie de actividades especificadas en la Figura 3., que sugiere un modelo de proceso para IWeb.

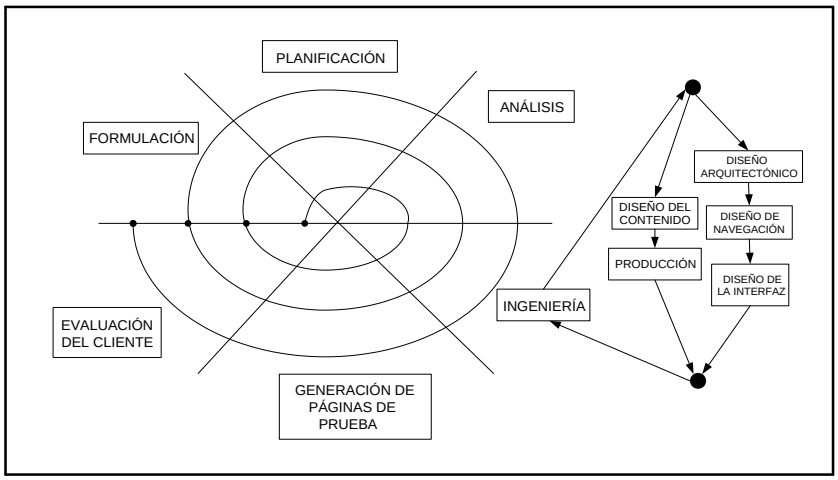

Figura 3. Modelo de Proceso IWeb.
Fuente: (Pressman, 2002, p.525)

El proceso IWEB empieza con *la formulación,* actividad que identifica las metas y los objetivos de la aplicación Web y establece el ámbito del primer incremento.

La planificación estima el costo global del proyecto, evalúa los riesgos asociados con el esfuerzo del desarrollo, y define una planificación del desarrollo bien granulada para el incremento final de la aplicación Web, con una planificación más toscamente granulada para los incrementos subsiguientes.

El *análisis* establece los requisitos técnicos para la aplicación Web e identifica los elementos del contenido que se van a incorporar. También se definen los requisitos

del diseño gráfico (estética). Adicionalmente, se exploran las limitaciones y el estado actual de los trabajos de investigación, comparándolos con las prácticas de investigación comercialmente aceptadas. Se discute la evidencia subjetiva que apoye el proceso de investigación iterativo, incorporando reuniones cliente-usuario para facilitar el desarrollo del entendimiento de las necesidades del usuario.

La actividad de *ingeniería* incorpora dos tareas paralelas, como se muestra a la derecha en la Figura 4. El *diseño del contenido* y la *producción* son tareas llevadas a cabo por personas no técnicas del equipo IWeb. El objetivo de estas tareas es diseñar, producir, y adquirir todo el contenido del texto, gráfico y vídeo que se vayan a integrar en la aplicación Web. Al mismo tiempo se lleva a cabo un conjunto de tareas de diseño.

La generación de páginas es una actividad de construcción que hace mucho uso de las herramientas automatizadas para la creación de la aplicación Web. El contenido definido en la actividad de ingeniería se fusiona con los diseños arquitectónicos, de navegación y de la interfaz paras elaborar páginas Web ejecutables en HTML, XML y otros lenguajes orientados a procesos (por ejemplo, Java). Durante esta actividad también se lleva a cabo la integración con el software intermedio *(middleware)* de componentes (es decir, CORBA, DCOM o JavaBeans).

Las pruebas ejercitan la navegación, intentan descubrir los errores de las *applets*, guiones y formularios, y ayuda a asegurar que la aplicación Web funcionará correctamente en diferentes entornos (por ejemplo, diferentes navegadores).

Cada incremento producido como parte del proceso IWeb se revisa durante la actividad de *evaluación del cliente.* Es en este punto en donde se solicitan cambios (tienen lugar ampliaciones del ámbito). Estos cambios se integran en la siguiente ruta mediante el flujo incremental del proceso (Pressman, 2002, pp.525-526).

Murugesan & Ginige (2005, p.11), basados en su experiencia practica en la construcción de aplicaciones Web, recomiendan una método de planificación de un proyecto en la Web con un proceso evolutivo para su desarrollo, mostrado en la Figura 4. Este proceso asiste a los desarrolladores en el entendimiento del contexto en el que la aplicación será desplegada y utilizada, ayuda a capturar los requerimientos, permite la integración de las diferentes disciplinas implicadas en este proceso, facilita la comunicación entre los miembros de desarrolladores, soporta la continua evolución y el mantenimiento, facilita la administración del contenido de información, y permite un manejo exitoso de la complejidad y diversidad del proceso de desarrollo(Ginige & Murugesan, 2001c, citado en Murugesan & Ginige, 2005, p.11).

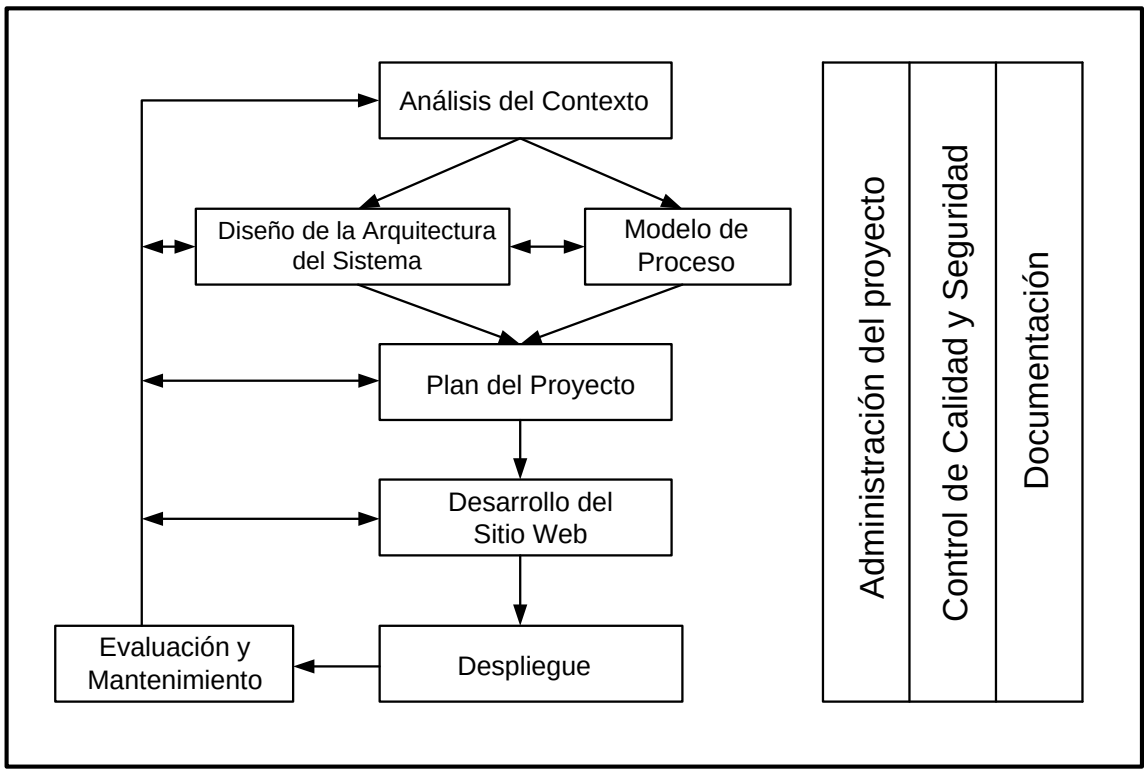

Figura 4: Proceso de Desarrollo de Proyectos Web
Fuente: (Muregesan & Ginige, 2005)

El primer paso esencial en el desarrollo de sistemas basados en Web, según este modelo, es el *análisis de contexto* en donde se obtiene y entiende los más importantes objetivos y requerimientos, así como las necesidades de los usuarios típicos del sistema y la organización que necesita el sistema. Es importante entender en esta etapa que los requerimientos pueden cambiar y evolucionar, incluso durante el desarrollo del sistema y después de su entrega. También es importante estudiar las operaciones por las que la aplicación esta siendo desarrollada, y las implicaciones potenciales de la introducción de nuevos sistemas en la organización.

Este estudio normalmente incluye: Cómo la información es creada y manejada; políticas organizacionales sobre pertenencia y control de la información, los planes actuales y futuros y los objetivos del negocio, el posible impacto de la introducción de estas aplicaciones en la organización; los cambios resultantes en el proceso de negocios, y las tendencias emergentes en la industria del sector (Murugesan & Ginige, 2005, pp. 11-13).

Tabla 3: Objetivos del análisis de contexto de las aplicaciones Web
Fuente: Murugesan & Ginige (2005, p.13)

Los objetivos del análisis de contexto, el primer paso en el desarrollo Web, son para:

- Identificar a las personas o grupos de interés (*stakeholders*) y sus amplios requerimientos y experiencia.
- Identificar las funciones que el sitio Web necesita proveer (inmediatamente, y a corto, mediano y largo plazo).
- Establecer qué información se necesita en el sitio Web, cómo conseguirla, y cuán seguido esta información puede cambiar.
- Identificar los requerimientos corporativos en relación con el *look and feel*, ejecución, seguridad, y gobernación.
- Estimar la cantidad de usuarios (típica y pico) y anticipar las demandas sobre el sistema.
- Estudiar similares (competitivos) sitios Web para ganar un entendimiento de sus funcionalidades, fortalezas y limitaciones.

En el *diseño de la arquitectura*, se decide sobre varios componentes del sistema y cómo es que deben estar conectados (Murugesan & Ginige, 2005, pp.13-14). En ésta etapa se diseña:

Una arquitectura global del sistema describiendo cómo la red y los variados servicios (servidores Web, servidores de aplicaciones y servidores de bases de datos) interactúan;

Una arquitectura de la aplicación representando varios módulos de información y de las funciones que soporta; y

Una arquitectura de software identificando variados tipos de software y módulos de bases de datos requeridos para implementar la arquitectura de la aflicción.

La Tabla 4 resume los medios para resaltar algunos de los requerimientos de las aplicaciones basadas en Web.

Tabla 4: Medios para realzar los requerimientos de las aplicaciones Web. Fuente: Ginige & Murugesan (2001c, incluido en Murugesan & Ginige, 2005).

Requerimientos	Medios para Resaltar
Look and feel uniforme a lo largo de todas las páginas Web que puedan ser fácilmente modificadas.	Creación de páginas Web usando *templates* y hojas de estilo.
Consistencia de la información que pueda aparecer en diferentes lugares o páginas.	Almacenar información en un solo lugar, como una base de datos, sin duplicación de la información en diferentes lugares o bases de datos, rescatando la información requerida para la presentación donde y cuando se necesite.
Fácil actualización y mantenimiento.	Suministro de un sistema secundario, (*back-end)* para editar la información en un repositorio de datos, podría tener interfaces Web para facilitar el acceso por cualquier lado.
Habilidad para adicionar nuevas páginas fácilmente	Generación dinámica de enlaces de navegación, antes que predeterminados enlaces de navegación estáticos.
Administración del sistema descentralizado.	Suministro de sistemas de suscripción, *(login)* a multi-usuarios para acceder al sistema secundario y la inclusión de un "sistema de administración de usuarios" que pueda asignar funciones específicas y datos a administradores de contenido y otros desarrolladores/administradores.
Mecanismo para el control de calidad facilitando la relevancia de la información.	Inclusión de metadatos o páginas Web. Uso de un robot Web para recolectar información silente, procesar la información recolectada y tomar la acción(es) apropiada para asegurar la calidad.
Incrementar la probabilidad de ser encontrado a través de motores de búsqueda.	Usar meta etiquetas *(meta tags)* y registrarlas con motores de búsqueda

Entonces se decide el *modelo de proceso* apropiado y se desarrolla un *plan de proyecto*. Para manejar un desarrollo Web exitoso, un plan de proyecto sólido y un cronograma realista son necesarios. El progreso de las actividades de *desarrollo del sitio* debe ser monitoreado y administrado. Las técnicas para planificar el proyecto y el cronograma que son comúnmente usadas en otras disciplinas pueden ser usadas para el desarrollo Web. A continuación, los componentes variados del sistema y las páginas Web son diseñados, desarrollados y probados.

El *despliegue* del producto una vez terminado es la etapa en la que muchos desarrolladores piensan que si tarea ha finalizado, sin embargo es entonces donde la complejidad de la aplicación se observa realmente.

Después de que un sistema basado en Web es desarrollado y desplegado en línea para su uso, necesita ser mantenido, Como se menciona anteriormente, el *mantenimiento* del contenido es un proceso continuo. Se necesita formular políticas y procedimientos de mantenimiento, basadas en las decisiones tomadas en la etapa de diseño de la arquitectura del sistema de cómo el contenido de la información sería mantenido, y luego implementar el plan. Además, como los requerimientos de los sistemas Web evolucionan y crecen con el tiempo, el sistema necesita ser actualizado y también posiblemente rediseñado para encajar con los nuevos requerimientos. Es importante la *evaluación* y revisión periódica de los sistemas basados en Web respecto al contenido de la información actual, los potenciales riesgos de la seguridad, la ejecución del sistema, los parámetros de uso (Murugesan & Ginige, 2005, pp.15-16).

4.3. Mantenimiento y Evolución de las Aplicaciones Web

Después de que un sistema es desarrollado y desplegado en línea para ser usado, es necesario que sea mantenido. El mantenimiento del contenido es un proceso continuo. Se necesitan formular políticas de mantenimiento, basadas en las decisiones tomadas en la etapa de diseño de la arquitectura. Estas decisiones determinan cómo la información del contenido crecerá y evolucionará, además de las necesidades a ser actualizadas y también cómo pueden ser rediseñadas para satisfacer los nuevos requerimientos. Es importante también revisar constantemente las aplicaciones y sistemas basados en Web con respecto a la información del contenido actual, los potenciales riesgos a la seguridad, el desempeño del sistema y los parámetros de uso. Además de medir adecuadamente los defectos y debilidades, si existiesen, para ser reparados (Murugesan & Ginige, 2005, pp.15-16).

Todos los problemas conocidos que caracterizan a la evolución de sistemas de software son exacerbados en el caso de las aplicaciones Web, porque su típico ciclo de vida es mucho más corto y sus iteraciones de desarrollo son ejecutadas más rápido y más frecuentemente. De hecho, las aplicaciones Web deben acomodarse continuamente a requerimientos cambiantes, ya que las necesidades de los usuarios evolucionan rápidamente al transcurrir del tiempo. Además la infraestructura tecnológica es en sí misma un tema de continua actualización. Las nuevas tecnologías emergen a un gran ritmo, reemplazando las existentes (Ricca & Tonella, 2005, p.243).

La resolución de los problemas anteriormente mencionados vendría a ser tarea del mantenimiento de las aplicaciones Web. Si bien el mantenimiento, generalmente se constituye en uno de los últimos pasos de cualquier proceso de desarrollo no es el menos importante, de hecho es un paso del que se depende mucho que su ejecución se lleve a cabo de forma correcta para el éxito de la aplicación (Ruiz & Polo, 2001).

En la definición de mantenimiento aparecen indicados, directa o indirectamente, cuatro tipos de mantenimiento (Ruiz & Polo, 2001):

Mantenimiento correctivo: es el conjunto de actividades dedicadas a corregir defectos en el hardware o en el software detectados por los usuarios durante la explotación del sistema

Mantenimiento preventivo: conjunto de actividades para facilitar el mantenimiento futuro del sistema

Mantenimiento perfectivo: conjunto de actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario

Mantenimiento adaptativo: es el conjunto de actividades para adaptar el sistema a los cambios (hardware o software) en su entorno tecnológico

Cuando la evolución de las aplicaciones Web hace que sus recursos y su código sean difíciles de mantener, la acción típica que es tomada consiste en reescribir la aplicación completa. Sin embargo, la posibilidad de (parcialmente) automatizar la reconstrucción puede ser efectiva y reducir los costos. Esta adopción envolvería la intervención periódica de mantenimiento preventivo para evitar la degradación progresiva (Ricca & Tonella, 2005, p.243). Por otro lado, para los fines de evolución de la aplicación Web un plan de mantenimiento adaptativo es el más recomendable para adaptar los nuevos requerimientos que surjan.

4.4. Pruebas y Evaluación

El proceso de prueba juega un rol crucial en el proceso de desarrollo general. Sin embargo, las pruebas y la evaluación son aspectos descuidados en el desarrollo Web. Muchos desarrolladores prueban el sistema solo después de producirse fallas. Lo que se desea en principio son las pruebas preactivas en varias etapas del desarrollo. Los beneficios del probado proactivo incluyen el aseguramiento del apropiado funcionamiento y los niveles de ejecución garantizados, evitar las reparaciones retroactivas costosas, optimizar la ejecución y disminuir el riesgo. Las pruebas y validación de grandes y complejos sistemas Web son tareas dificultosas y costosas. No debería ser una tarea guardada para el final del proceso de desarrollo. Un plan para realizar pruebas necesita ser concebido en etapas tempranas del ciclo de vida. De ser así se podrá contar con mapas, entonces el sitio Web podrá ser evaluado a través de las etapas de establecimiento de requerimientos y diseño. También ayuda a estimar el tiempo de esfuerzo necesitado para concretar las tareas de prueba. Las categorías siguientes proveen excelentes guías práctica de cómo se debe evaluar un sistema Web (Murugesan & Ginige, 2005, pp.15-16):

Compatibilidad con el Browser
Despliegue de páginas
Administración de sesiones

Usabilidad
Análisis de contenido
Disponibilidad
Respaldo y recuperación
Transacciones
Compras, procesamiento de órdenes
Internacionalización
Procedimientos de negocios operacionales
Integración de sistemas
Ejecución
Registro de usuarios y seguridad

La experiencia muestra que si bien las pruebas y la evaluación llegan a ser tareas costosas el impacto resultante de las fallas debido a la falta de pruebas podría ser más costoso o incluso desastroso.

4.4.1. Tratamiento de la Calidad en las Aplicaciones Web

Se espera que un producto de software, ya sea centrado en la Web o tradicional, por lo menos funcione. Sin embargo, las expectativas ideales que se tiene de un producto de software van más allá del simple funcionamiento, y suponen un aprovechamiento máximo de los recursos, eficiencia en el funcionamiento, facilidad en el mantenimiento, y tendencia a la evolución.

Procurar todas esas cualidades en un producto de software suele no ser una tarea sencilla, y en el caso particular de las aplicaciones Web el panorama se complica más, por el hecho de que su estudio es relativamente reciente. Afortunadamente, existen estándares de evaluación e implementación de calidad de las aplicaciones Web que permiten hacer de este ideal algo más cercano a la realidad y así poder corregir, adaptar, mejorar y soportar la aplicación a largo plazo.

El tratamiento de la calidad también llega a ser una labor compleja debido a la diversidad de criterios existentes para definir a un objeto como poseedor o no poseedor de calidad. Para los usuarios finales la percepción de la calidad varía enormemente de un sujeto a otro. Algunos usuarios disfrutan con gráficos llamativos, en cambio otros solo quieren un texto sencillo. Algunos exigen información copiosa, otros desean una presentación abreviada. Para los desarrolladores el poder distinguir los criterios de calidad que se ajusten a la mayoría de las necesidades es de vital importancia, ya que, la percepción de lo bueno, por parte del usuario podría ser más importante que cualquier discusión técnica sobre la calidad de las aplicaciones Web (Pressman, 2002, p.524).

Estas situaciones indican la necesidad de encontrar un equilibrio entre lo que se cree que es un atributo de calidad y lo que en la práctica lo es realmente. En la tesis de Olsina (1999) se tratan dos conceptos importantes y extremos -aunque no excluyentes en la práctica- la: objetividad y subjetividad; y que, en el contexto de la

evaluación significa el proceso de decisión basado en modelos y métodos cuantitativos, o al proceso de decisión basado en la intuición. Además, cuando se habla de intuición se refiere al mecanismo mental, o conocimiento claro, íntimo o instantáneo, surgido de la experticia. Con todo, es importante resaltar que la evaluación intuitiva es un ejercicio mental del día a día que todo ser humano realiza, ya sea más o menos experto en el domino del objeto que evalúa. Por otra parte, más de un investigador ha afirmado que la métrica de un atributo tiene que estar bien definida, esto es, basada en la observación empírica, consistente con la intuición y que represente una apropiada correspondencia entre el dominio empírico y el sistema numérico.

Lo anterior simplemente muestra que el uso de estándares y modelos de calidad es incompleto si no se cuenta con la experiencia y el análisis intuitivo que le indique a la persona encargada de realizar esta tarea la manera de evaluar, aplicar e implementar la calidad en las aplicaciones Web. La evaluación de sistemas complejos, en este caso particular de sistemas Web es primeramente un problema de decisión. Consecuentemente, las estrategias y técnicas para su solución deben contener componentes subjetivos. Esto se afirma en las siguientes sentencias: "El uso de métricas de software reduce la subjetividad en la evaluación de la calidad de software al proveer una base cuantitativa para tomar decisiones acerca de la calidad del software" (IEEE Std 1061, citado en Olsina, 1999). "No obstante, el uso de métricas de software no elimina la necesidad del juicio humano en la evaluación del software" IEEE Std 1061, citado en Olsina, 1999)

4.4.2. Fases del Proceso de Evaluación Web

Se considera que la aplicación estudiada es compleja desde el punto de vista de la evaluación, ya que a la calidad se la expresa en función de varias características (en consideración de un perfil de usuario), y que las mismas se descomponen recursivamente en subcaracterísticas y atributos. Entonces, ¿cuáles las principales fases o módulos del proceso de evaluación de una aplicación Web? Se puede argumentar las siguientes fases:

Con respecto a la fase de *Planificación y Programación de la Evaluación de Calidad*, la misma contiene actividades y procedimientos de soporte, con el fin de determinar objetivos estratégicos, tácticos y operativos. Esto es, permite establecer las principales estrategias y metas del proceso en un contexto organizacional; permite seleccionar un modelo de proceso de evaluación, asignar métodos, agentes y recursos a las actividades; programar y re-planificar una vez en marcha el proceso de evaluación.

Considerando a la fase de *Definición y Especificación de Requerimientos de Calidad,* la misma trata con actividades y modelos para la determinación, análisis y especificación de los requerimientos. A partir de un proceso de medición orientado a metas, y con el fin de evaluar, comparar, analizar, y mejorar características y atributos de artefactos Web, los requerimientos deben responder a necesidades y comportamientos de un perfil de usuario y dominio dados. El proceso de

determinación de requerimientos, realizado en una mezcla de estrategias prescriptivas y descriptivas, culmina con un documento que jerárquicamente especifica a todas las características y atributos cuantificables que modelan a la calidad según las necesidades del usuario.

Con respecto a la fase de *Definición e Implementación de la Evaluación Elemental* la misma trata con actividades, modelos, técnicas y herramientas para determinar métricas y criterios de evaluación para cada atributo cuantificable. Se consideran tipos de criterios elementales, escalas, escalas de preferencia, valores críticos, y funciones para determinar la preferencia elemental, entre otros asuntos. Una vez definidos y consensuados los criterios para medir cada atributo, se debe ejecutar el proceso de recolección de datos, computar las métricas y preferencias elementales, y documentar los resultados.

Considerando a la fase *Definición e Implementación de la Evaluación Global* la misma trata con actividades, modelos, y herramientas para determinar los criterios de agregación de las preferencias de calidad elemental para producir la preferencia global, para cada sistema seleccionado. Se consideran tipos de funciones de agregación para modelar diferentes relaciones entre atributos y características, a saber: relaciones de reemplazabilidad, simultaneidad, neutralidad y diferentes niveles de polarización. Una vez definidos y consensuados los criterios, se debe llevar a cabo el proceso de cálculo y *ranquin*.

Con respecto a la fase de *Análisis de Resultados, Conclusiones y Documentación* la misma trata con actividades de análisis y comparación de las preferencias de calidad elementales, parciales y globales, y, asimismo, la justificación de los resultados. Por otra parte, se utilizan herramientas y mecanismos de documentación para facilitar la interpretación de los datos y su seguimiento (Olsina, 1999).

4.4.3. Planificación de los Criterios de Calidad en La Web

La evaluación de las aplicaciones Web es necesaria realizarla de manera cuantitativa y cualitativa. El elevado número de páginas existentes obliga a contar con criterios desde los que se extraiga la información de calidad de la abultada cifra de recursos inservibles, inoperantes y desdeñables. Asimismo, cualquier fuente de información sólo es válida si aporta contenidos útiles y si los mismos son localizados de forma sencilla. Por este motivo, también es necesario recurrir a parámetros que ayuden a identificar la información imprescindible y separarla de la que nada aporta. Es evidente que disponer de indicadores para aplicar en el proceso de evaluación es, sin lugar a dudas, necesario, así como se detalla en Merlo (2003).

La evaluación de de una aplicación Web, requiere una planificación concreta en la que se establecerán los criterios que se aplicarán y los métodos mediante los que se pondrán en práctica dichos criterios. Los criterios se materializarán mediante el uso de parámetros e indicadores de evaluación; mientras que los métodos se desarrollan a través de procedimientos concretos y la ayuda de los recursos necesarios para la realización positiva de los métodos ideados para llevar a cabo el proceso de

evaluación Parámetros, indicadores, procedimientos y recursos son, por tanto, los cuatro elementos clave del proceso de evaluación de la información Web.

Los *parámetros* son los aspectos genéricos que serán evaluados. Se trata de establecer una serie de grandes bloques sobre los que se realizará el análisis y los cuales serán desarrollados en indicadores concretos que dan la información necesaria para cada uno de estos grupos.

Los *indicadores* son los elementos que desarrollan cada uno de los parámetros establecidos para el análisis de la información. Son las cuestiones concretas que se evaluarán. Como ocurre con los parámetros, existen múltiples componentes que pueden ser considerados como un índice de la calidad de una página o de un sitio Web.

Los *procedimientos* son los métodos que se emplean para hacer efectiva la aplicación de parámetros e indicadores. Este es el aspecto del proceso de evaluación que presenta un menor grado de desarrollo en cuanto a aportaciones teóricas o experiencias prácticas, ya que sólo hay propuestas aisladas y parciales. La planificación de cualquier proceso de evaluación no puede limitarse a delimitar qué se debe analizar sino también debe decir cómo se debe obtener la información relativa a los elementos que se están evaluando. La evaluación de información Web adolece, en estos momentos, de una definición y sistematización de los procedimientos que se deberán aplicar.

Los *recursos* son los materiales necesarios para el proceso de evaluación. Conocidos qué aspectos serán evaluados y cómo se procederá a su análisis será necesario establecer qué medios humanos, instrumentales y documentales son necesarios. Como ocurría con los procedimientos, los recursos también están poco estructurados y, por lo general, en la planificación y ejecución de la evaluación, sólo se contemplan los recursos humanos y algunos documentales como las listas de parámetros e indicadores y los formularios o plantillas de análisis. Sin embargo, un correcto proceso de análisis de la calidad de la información Web debería dejar constancia, antes de comenzar con la evaluación, de todos los recursos que serán necesarios para el desarrollo del mismo.

4.4.4. Criterios de Calidad en la Web

Una vez habiendo establecido las fases del proceso de la evaluación Web y la planificación de los criterios de evaluación de calidad, es el momento de conocer los criterios de calidad que mejor se ajustan al proceso de evaluación y aplicación de calidad en la Web. Existe un elevado número de aportaciones en cuanto a los parámetros e indicadores que se deben aplicar para la evaluación. Olsina (1999) y sus colaboradores, la Organización de Estándares Internacionales ISO, en trabajo conjunto con IEC, definen seis características de muy alto nivel que describen, con un mínimo de solapamiento, a la calidad del producto y que son, a saber: usabilidad, funcionalidad, confiabilidad, eficiencia, portabilidad, y mantenibilidad. “Esas

características proveen una línea base para ulteriores refinamientos y descripciones de la calidad del software".

A continuación se presenta un "árbol de requisitos de calidad" Figura 5 que identifica un conjunto de atributos que conduce a aplicaciones Web de alta calidad (Pressman, 2002).

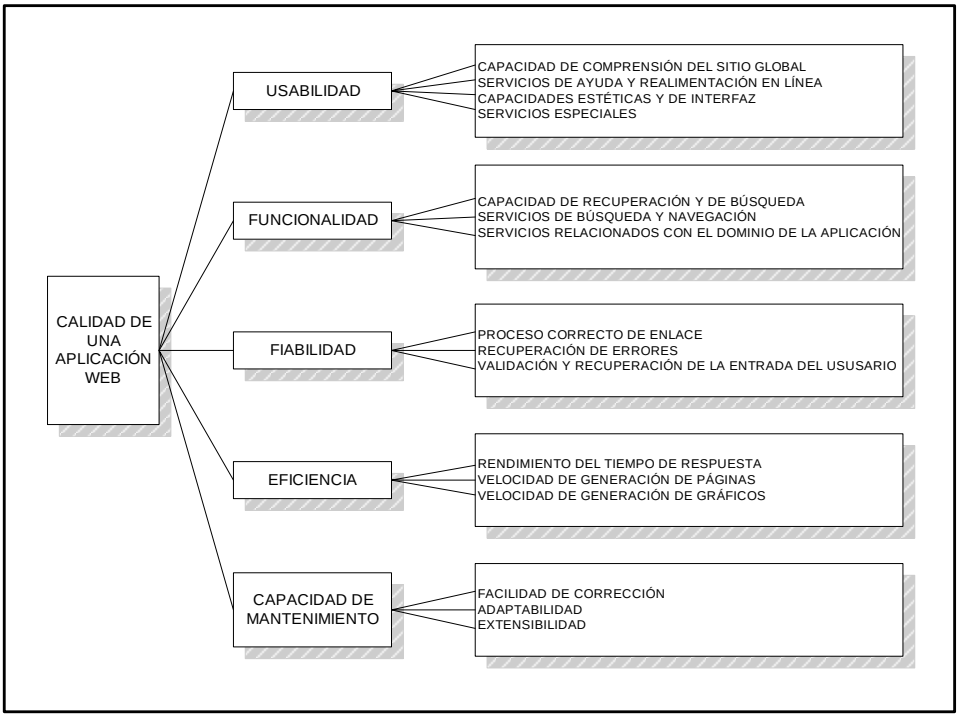

Figura 5: Árbol de requisitos de calidad

Fuente: Olsina (1999) incluido en Pressman (2002).

4.5. Pasos para el Éxito del Desarrollo Web

El *éxito en el desarrollo Web* envuelve múltiples pasos interactivos que influyen uno sobre otro. Murugesan & Ginige (2005) recomiendan los siguientes pasos claves para el desarrollo y despliegue de las aplicaciones Web:

> Entender el ambiente de funcionamiento y operaciones generales del sistema incluyendo los objetivos de negocios y requerimientos, cultura de organización y las políticas de manejo de la información.
>
> Identificar claramente a los stakeholders, que son los usuarios principales del sistema y sus perfiles típicos; a la organización que necesita el sistema; y a los que patrocinan el desarrollo.
>
> Obtener o especificar los requerimientos (inicialmente) funcionales, técnicos, y no técnicos de los stakeholders y del sistema general. Además, reconocer

que estos requerimientos pueden no permanecer igual; si no que pueden evolucionar con el tiempo.
Desarrollar la arquitectura general del sistema basado en Web que conozca los requerimientos técnicos y no técnicos.

Identificar subprojectos o subprocesos para implementar la arquitectura del sistema. Si los subproyectos son muy complicados de manejar, más adelante dividirlos hasta convertirlos en un conjunto de tareas manejables.

Desarrollar e implementar los subproyectos.

Incorporar mecanismos efectivos para manejar la evolución del sistema, sus cambios y su mantenimiento. A medida que el sistema evoluciona, repetir el proceso general o algunas partes así como se requiera.

Dirigir los aspectos no técnicos, tales como procesos de negocios revisados, las políticas de organización y administración, desarrollo de recursos humanos, y aspectos legales, culturales y sociales.

Medir la ejecución del sistema, analizar el uso de la aplicación Web con registros Web, y revisar y dirigir las opiniones y sugerencias de los usuarios.

Refinar y actualizar el sistema

4.6. Conocimiento y Aptitudes para el Desarrollo Web

El conocimiento y las aptitudes necesarias para el desarrollo de grandes y complejas aplicaciones Web son muy diversas y abarcan muchas disciplinas diferentes. En términos generales se pueden clasificar de la siguiente forma (Murugesan & Ginige, 2005, p.20):

Las tecnologías que soportan y facilitan el desarrollo Web.
Métodos de diseño.

- Diseño para la navegación –diseño de interfase y navegación.
- Diseño para la comprensión.
- Diseño de la ejecución –capacidad de respuesta.
- Diseño para la seguridad e integridad.
- Diseño para la evolución, crecimiento y mantenibilidad.
- Diseño para las pruebas.
- Gráficos y diseño multimedia.
- Desarrollo de páginas Web

Arquitectura de sistemas
Métodos y procesos de desarrollo Web
Administración de proyectos Web

Desarrollo de herramientas
Administración del contenido
Requerimientos de estándares y regulaciones
Análisis y métricas de calidad

4.7. Equipo de desarrollo Web

Como se menciona anteriormente, el desarrollo de de aplicaciones Web requiere de un equipo de personas con diferentes destrezas y conocimientos (Hansen, 2004, citado es Murugesan & Ginige, 2005). Estos individuos incluyen programadores, diseñadores gráficos, diseñadores de páginas Web, expertos en usabilidad, desarrolladores de contenido, diseñadores y administradores de bases de datos, expertos en comunicación de datos y redes, y administradores de servicios Web. Un equipo de desarrollo Web es multidisciplinario, como un equipo de producción de películas, y debe ser más versátil que un equipo de desarrolladores de software tradicional. Hansen et al. (2001, citado en Murugesan & Ginige, 2005) presentan una clasificación de los participantes del equipo de desarrollo Web y una jerarquía por sus destrezas y conocimientos. Esta clasificación ayuda a formar un equipo y a concebir una estrategia para el equipo de desarrolladores.

5. Metodología UWE (UML-Based Web Engineering)

Existe una gran variedad de metodologías específicamente desarrollados para la construcción de aplicaciones Web. Las metodologías toman un papel importante y riguroso en cuanto a su desarrollo por lo mismo se requiere un conocimiento de las diferentes metodologías que abarque todo el ciclo de vida de las aplicaciones Web. La necesidad del tratamiento adecuado de la navegación ha sido una de las características que ha llevado a diversos grupos de investigación a proponer nuevos modelos y técnicas adecuadas para su tratamiento.

Todas las metodologías presentan su propia notación y tipos de diagramas, y contribuyen con ideas importantes para el diseño de software basado en Web. El Lenguaje de Modelado Unificado, *Unified Modeling Language (UML)* es una herramienta lo suficientemente poderosa para cubrir todos los requerimientos que surgen cuando se modela una aplicación Web (Koch & Kraus, 2002). Además, tiene la ventaja de ser un lenguaje de modelado bien documentado, que es de hecho un estándar industrial y la notación orientada a objetos más utilizada en la actualidad. Por lo tanto, permite a cualquier practicante con una noción general de UML ser capaz de entender un modelo basado en esta especialización (Koch, Kraus & Hennicker, 2001).

El enfoque *UWE (UML-Based Web Engineering),* presentado por Koch y sus colegas y extendida en subsecuentes artículos, soporta el desarrollo de aplicaciones Web con especial énfasis en sistematización, personalización y semi-automática generación. Está fundada en aproximaciones orientadas a objetos, iterativas e incrementales basándose en dos principales dimensiones, es decir el tiempo y el contenido, del

proceso unificado. Para la notación se hace uso de un "ligero" (lightweight) perfil de UML. Este perfil presenta *estereotipos* definidos por el modelado de aspectos de navegación y presentación de aplicaciones Web. UWE proporciona guías para la construcción de modelos de forma sistemática y con pasos acertados, enfocadas en personalización y en estudio de casos de uso. El método recomienda el uso de restricciones escritas en un lenguaje de restricciones OCL para aumentar la precisión de los modelos. Dentro de UML, OCL es el estándar para la especificación de invariantes de clases y operaciones de pre-condiciones y pos-condiciones (Koch & Kraus, 2002).

La sintaxis y la semántica de estos elementos de modelado son definidas por un *meta-modelo de UML* y de las *reglas bien formadas.* También se consideran mecanismos de extensión incorporados que son los *estereotipos (stereotypes)*, *valores etiquetados (tagged values)* y *restricciones (constraints)*. UML puede ser visto como una familia de lenguajes de modelado, más que un lenguaje simple (Cook (2000) citado por Koch & Kraus (2002)) si se consideran las extensiones "ligeras". Por "ligero", se entiende que puede ser soportado fácilmente por herramientas y que no significa gran impacto en el intercambio de formatos. Los *estereotipos* son un nuevo tipo de elementos de modelado definidos dentro del modelo basado en un tipo de elementos de un modelo existente. Un *valor etiquetado* es un par (etiqueta, valor) que permite adjuntar información arbitraria a cualquier elemento de modelado. Una *restricción* es una condición o limitación que permite nuevas semánticas para ser especificadas de forma lingüística para un elemento de modelado (Koch & Kraus, 2002).

UWE proporciona guías para la construcción de modelos de forma sistemática y con pasos acertados, enfocadas en personalización y en estudio de casos de uso. Las actividades de modelado principales son el *análisis de requerimientos, el diseño conceptual, el diseño de navegación y el diseño de presentación*, y producen los siguientes artefactos (Koch et al., 2001; Hennicker & Koch 2001; Koch & Kraus, 2002):

Modelo de Casos de Uso

Modelo Conceptual

Modelo de Espacio de Navegación y Modelo de estructura de Navegación

Modelo de presentación

5.1. Análisis de Requerimientos con Casos de Uso

El objetivo del *análisis de requerimientos* es cumplir las tareas elicitación de requisitos, definir y validar los requerimientos de los usuarios de la aplicación Web. Se hace una distinción entre requerimientos funcionales y no funcionales. Los requerimientos funcionales tratados en UWE son (Escalona & Koch, 2003):

Requerimientos relacionados con el *contenido*.

Requerimientos relacionados con la *estructura*.

Requerimientos relacionados con la *presentación*.

Requerimientos relacionados con la *adaptación*.

Requerimientos relacionados con el *usuario*.

Para describir los requerimientos funcionales de una aplicación se puede usar un *modelo de casos de uso* (Ver figura 6 y figura 7). Este modelo describe un trozo de comportamiento de la aplicación sin revelar su estructura interna. El modelo de casos uso está conformado por dos elementos de modelado principales, llamados casos de uso y actores. Un *caso de uso (use cases)* es una unidad coherente de funcionalidad provista de aplicaciones que interactúan con uno o más actores externos de la aplicación. Un *actor (actor)*, es el rol que un usuario puede desempeñar con respecto a un sistema o una entidad, tales como otro sistema o una base de datos. Además, existen relaciones de casos de uso entre estos dos elementos, tales como las *asociaciones (associations)* entre actores y casos de uso y las *dependencias (dependencies)* «includes» y «extends» entre casos de uso (Koch et al., 2001).

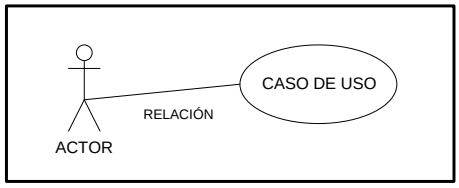

Figura 6: Elementos de un Modelo de Casos de Uso
Fuente: Modificado de Zuazo (2004)

El análisis de requerimientos es una técnica centrada en el usuario que obliga a definir quienes son los actores de la aplicación y ofrece un camino intuitivo de representar la funcionalidad que la aplicación tiene que satisfacer para cada actor (Koch et al., 2001).

Los pasos para dirigir este proceso de caos de uso son (Koch et al., 2001):

1. Identificar a los actores
2. Para cada actor identificar las actividades que desempeña
3. Agrupar las actividades en casos de uso
4. Establecer relaciones entre los actores y los casos de uso
5. Establecer relaciones de tipo «include» y «extend» entre los casos de uso.
6. Simplificar el modelo de casos de uso mediante la definición de relaciones de herencia entre actores y casos de uso

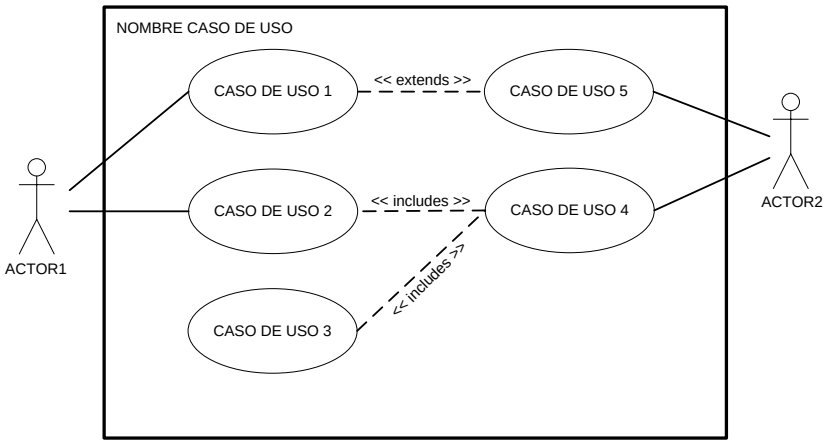

Figura 7: Modelo de Casos de Uso
Fuente: Modificado de Hennicker & Koch (2001) y Koch & Kraus (2002)

5.2. Representación del Modelo Conceptual

El diseño conceptual está basado en el análisis de requerimientos del paso previo. Incluye a los objetos involucrados en la interacción entre el usuario y la aplicación, especificado en los casos de uso. Apunta a la construcción de modelos de clase con estos objetos, que intentan ignorar tanto como sea posible los caminos de navegación y los pasos de presentación (Koch et al., 2001).

Cuando se desarrolla un análisis orientado a objeto, se emplea una búsqueda de conceptos u objetos, y no como en las metodologías estructuradas, donde se buscan funciones; por lo que es prudente buscar la mayor cantidad de conceptos para desarrollar un modelo conceptual cercano a la realidad. Para ello se puede identificar las frases nominales en las descripciones textuales del dominio del problema y considerarlas conceptos idóneos, pero que se deben categorizar para tener una mejor comprensión de la realidad que engloba la problemática (Zuazo, 2004).

Los principales elementos usados para el modelo conceptual son las *clases (classes)* y *asociaciones (associations).* Sin embargo, el poder del diagrama de clases es dado por una variedad de características adicionales que pueden ser usadas para mejorar sistemáticamente estos diagramas. Ejemplos de estas características son los *nombres de asociación (named associations)* y *los nombres de roles de asociación (role names)*, la *cardinalidad (multiplicities)*, diferentes formas de asociaciones soportadas por UML como *agregación (aggregation)*, *herencia (inheritance)*, *composición (composition)* y *la clase asociación (association class),* todas estas representadas gráficamente utilizando notación de UML. Si el modelo conceptual consiste de varias clases se recomienda que se agrupen usando elementos de modelado de *paquetes (package)* de UML (Koch & Kraus, 2002).

Una *clase* es descrita por un *nombre, atributo, operaciones* y *variantes* como se ve en la Figura 8. El compartimiento opcional llamado variante puede ser añadido a las clases. Contiene información usada para la funcionalidad del *contenido adaptativo*.

Las *asociaciones* y *paquetes* son usadas como en los diagramas de clase estándares de UML (Koch et al., 2001).

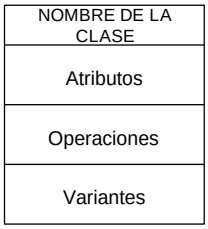

Figura 8: Clase con Variantes de Compartimiento Adicionales
Fuente: (Koch et al. 2001)

Como método para construir este modelo de clases (Ver figura 9) para el dominio se siguen técnicas de modelado tales como: (Koch et al., 2001)

1. Encontrar clases
2. Especificar los atributos y las operaciones más relevantes.
3. Determinar las asociaciones entre clases.
4. Agregar clases e identificar la composición de clases.
5. Definir las jerarquías de herencia
6. Definir las restricciones

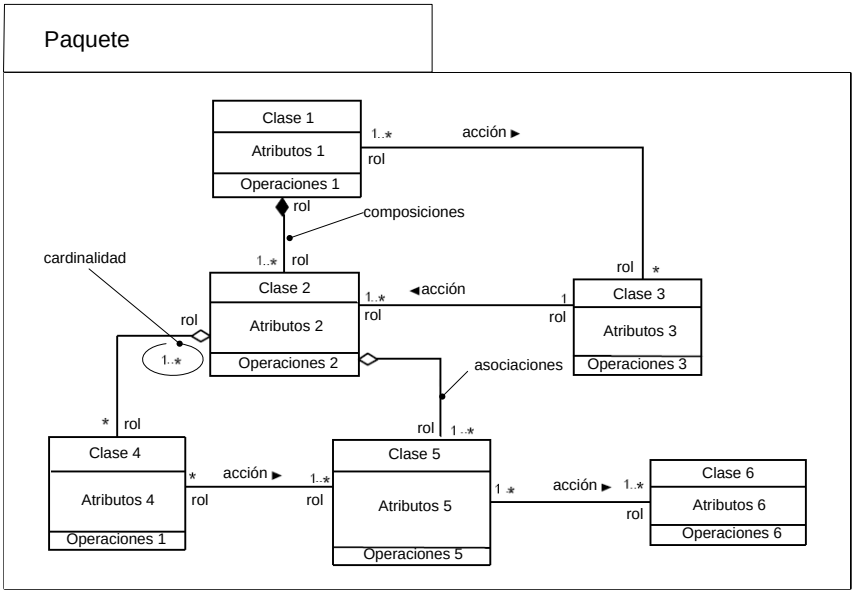

Figura 9: Modelo Conceptual
Fuente: Modificado de Koch & Kraus (2002) y Zuazo (2004)

5.3. Modelo de Navegación

El *diseño de navegación* es un paso crítico en el diseño de la aplicación Web. Por un lado, los enlaces aumentan la navegabilidad, por otro lado, sin embargo, incrementan el riesgo de perder la orientación. Construir un modelo de navegación

no sólo es de gran ayuda para la documentación de la estructura, también permite acceder a un crecimiento estructurado de la navegabilidad. Este modelo se comprime en el modelo de espacio de navegación y el modelo de estructura de navegación. El primero especifica *qué* objetos pueden ser visitados mediante una navegación a través de la aplicación. *Cómo* estos objetos son alcanzados está definido por el modelo de la estructura de navegación (Koch et al., 2001).

5.3.1. Modelo de Espacio de Navegación

En el proceso de construir el *modelo de espacio de navegación* el desarrollador toma decisiones cruciales de diseño, tales como qué vista del modelo conceptual es necesaria para la aplicación y cuáles serán los caminos de navegación requeridos para el aseguramiento de la funcionalidad. Las decisiones del diseñador están basadas en el modelo conceptual y en los requerimientos de la aplicación definidos en el modelo de casos de uso. Los elementos utilizados para este modelo son las *clases de navegación* y las *asociaciones de navegación*, que expresan la *navegación directa* (Koch et al., 2001).

La *clase de navegación (navegation class)* modela una clase cuyas instancias son visitadas por usuarios durante la navegación (Ver figura 10). Se les asigna el nombre que se diera a las correspondientes clases conceptuales. Sin embargo, se diferencia de esta por el estereotipo <<*navigation class*>>. Además, una clase de navegación puede contener atributos de otras clases del modelo conceptual, siempre que la clase de navegación tenga alguna asociación con la clase de la que se presta el o los atributos. Para diferenciar dichos atributos se coloca una barra inclinada a la derecha (/) antes del nombre (Koch et al., 2001).

La *navegación directa (direct navegability)* es representada por asociaciones en el modelo de espacio de navegación que provienen de la clase de navegación de origen. Por lo tanto, sus semánticas son diferentes de las asociaciones usadas en el modelo conceptual. . Estas asociaciones son interpretadas como el enlace o vínculo entre la clase de navegación inicial (página Web inicio) y la clase de navegación final (página Web destino). Se representan mediante una flecha unidireccional o bidireccional adjuntada a uno o ambos extremos de la asociación. Cada extremo dirigido se encuentra etiquetado con el nombre de un rol y la cardinalidad correspondiente. En caso que el nombre del rol no esté explícito, se utiliza la siguiente convención: si la cardinalidad es menor o igual a uno, el nombre de la clase destino es utilizado como el nombre del rol; si la cardinalidad es mayor que uno, la forma plural del nombre de la clase destino es utilizada como nombre del rol. El estereotipo utilizado para identificar a esta asociación es <<*direct navigability*>> (Koch et al., 2001).

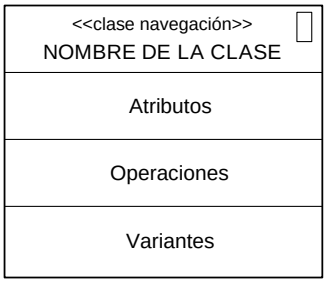

Figura 10: Clase Navegación
Fuente: (Koch et al., 2001)

El modelo de espacio de navegación es construido con las clases de navegación y las asociaciones de navegación y están representadas gráficamente por un diagrama de clases de UML. A pesar de que no hay un medio para automatizar la construcción del modelo de espacio de navegación se utilizan varias guías para ser seguidas por el desarrollador (Koch et al., 2001):

1. Incluir clases del modelo conceptual que son relevantes para la navegación como las clases de navegación. Si la clase conceptual no es un objetivo de visita en el modelo de casos de uso, es irrelevante en el proceso de navegación, y por ende omitido en el modelo.

2. Mantener la información de las clases omitidas (si fuera necesario) como atributos de otras clases en el modelo de espacio de navegación. Todos los otros atributos de clases de navegación se ponen en mapas directamente como atributos de la clase conceptual correspondiente. En cambio, excluir los atributos de las clases conceptuales que son consideradas irrelevantes para la presentación en el modelo.

3. Las asociaciones del modelo conceptual son mantenidas en el modelo de navegación. Las asociaciones pueden ser añadidas para la navegación directa y, de esta manera, evitar caminos de navegación demasiado largos.

4. Añadir asociaciones adicionales basadas en la descripción de los requerimientos o de los escenarios descritos por el modelo de casos de uso.

5. Añadir limitaciones para especificar restricciones en el espacio de navegación.

5.3.2. Modelo de Estructura de Navegación

El modelo de estructura de navegación describe cómo la navegación es soportada por elementos de acceso tales como índices, visitas guiadas, preguntas y menús. Técnicamente, los caminos de navegación junto con los elementos de acceso son representados por los modelos de clase que pueden ser sistemáticamente construidos del modelo de espacio de navegación en dos pasos: El primer paso consiste en

realzar el modelo de espacio de navegación con índices, visitas guiadas y preguntas. El segundo consiste en derivar menús directamente del modelo realzado. Los menús representan posibles elecciones de navegación. El resultado es un diagrama de clases UML construido con estereotipos UML, los cuales están definidos según mecanismos de extensión UML (Koch et al., 2001).

5.3.2.1. Primitivas de Acceso

Las primitivas de acceso son nodos de navegación adicionales requeridas para acceder a objetos de navegación. Las siguientes primitivas de acceso son definidas como estereotipos UML: índices, visitas guiadas, consultas y menús. Las tres primeras son descritas y usadas para refinar el modelo de espacio de navegación. Los menús son tratados separadamente (Koch et al., 2001).

Los *índices (index)* (ver figura 11) permiten el acceso directo a las instancias de la clase de navegación. Esto es modelado por un objeto compuesto, el que contiene un número arbitrario de ítems indexados. Cada ítem indexado está en torno a un objeto, el cual tiene un nombre que identifica la instancia y posee un enlace a una instancia de una clase de navegación. Cualquier índice es miembro de la clase índice, y utiliza el estereotipo <<*index*>> con su icono correspondiente (Koch et al., 2001).

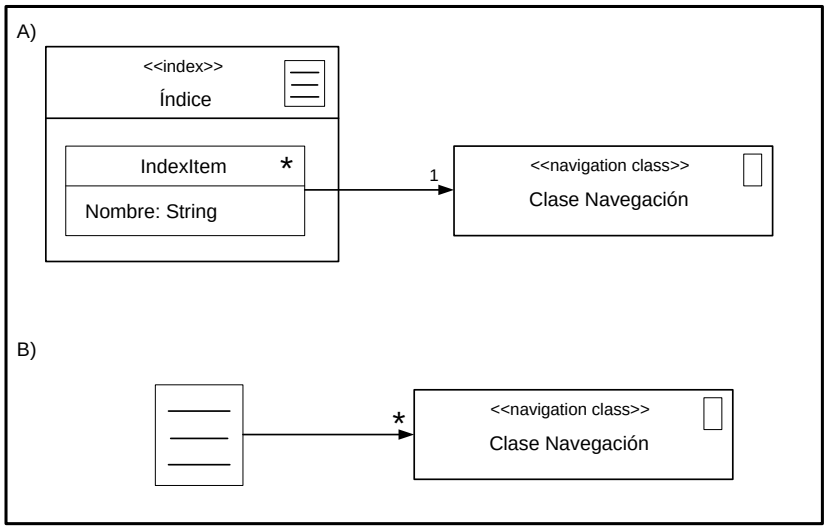

Figura 11: a) Clase Índice y b) su notación taquigráfica
Fuente: (Koch et al., 2001; Hennicker & Koch, 2001)

Las *visitas guiadas (guided tour)* (ver figura 12) proveen acceso secuencial a las instancias de una clase navegación. Para clases que contienen objetos de visita guiada se usa el estereotipo <<*guidedTour*>> y su icono correspondiente. Otros elementos llamados *NextItem* deben ser conectados a una clase navegación. Las visitas guiadas deben ser controladas por el usuario o por el sistema (Koch et al., 2001).

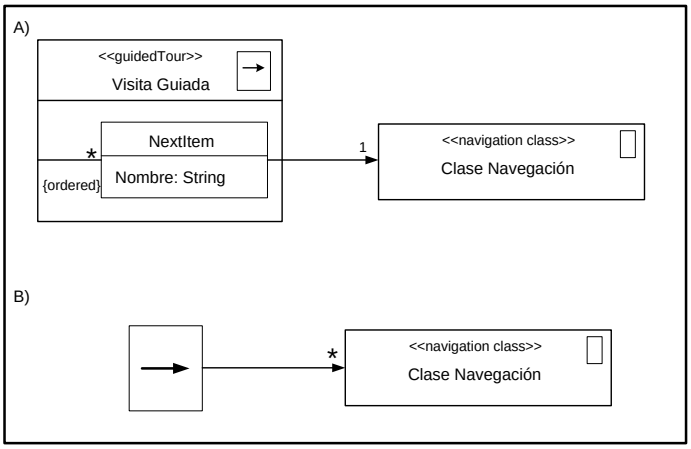

Figura 12: a) Clase Visita Guiada y b) su notación taquigráfica
Fuente: (Koch et al., 2001; Hennicker & Koch, 2001)

Una *consulta (query)* (ver figura 13) es modelada por una clase que tiene una serie de preguntas como atributo. Para la clase consulta se utiliza el estereotipo <<*query*>> y su icono correspondiente. Cualquier clase consulta es la fuente de dos asociaciones dirigidas relacionadas por la restricción *{xor}*. De esta forma una pregunta con varios objetos resultantes es modelada para llevar primero a un soporte índice la selección de una instancia particular de una clase navegación. Es resultado de las consultas puede alternativamente ser usada como entrada para una visita guiada (Koch et al., 2001).

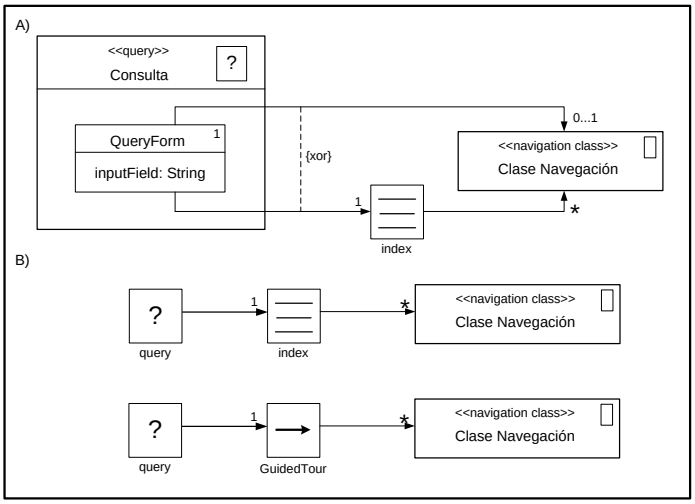

Figura 13: a) Clase Consulta y b) sus notaciones taquigráficas
Fuente: (Koch et al., 2001; Hennicker & Koch, 2001)

Para el modelado deben seguirse ciertas reglas que son resumidas a continuación (Koch et al., 2001):

1. Reemplazar todas las asociaciones bidireccionales que tengan cardinalidad mayor que uno en ambos extremos de la asociación por dos asociaciones unidireccionales correspondientes.

2. Reemplazar todas las asociaciones bidireccionales que tengan cardinalidad mayor que uno en un extremo de la asociación con una asociación unidireccional con un extremo dirigido de la asociación en el extremo con cardinalidad mayor que uno. La navegación en la otra dirección está garantizada por el uso de árboles de navegación introducidas más tarde en el diseño.

3. Considerar solo aquellas asociaciones del modelo de espacio de navegación, que tengan cardinalidad mayor que uno en el extremo dirigido de la asociación.

4. Para cada asociación de esta clase, escoger uno o más elementos de acceso para entender la navegación.

5. Resaltar el modelo de espacio de navegación correspondientemente. Los nombre de los roles en este modelo son ahora movidos hacia los elementos de acceso. Si dos o más alternativas son introducidas en el paso 3, distinguirlas mediante el cambio de nombres de roles de la asociación por medio de búsquedas o el criterio del índice usado.

5.3.2.2 Adición de Menús

En este paso, las primitivas de acceso de tipo menú son añadidas al modelo de estructura de navegación.

El elemento de modelado *menú* (ver figura 14) es una primitiva de acceso adicional que puede ser añadida a la lista presentada en el paso previo. Un menú es un índice de un conjunto de elementos heterogéneos, tales como índices, visitas guiadas, consultas, una instancia de una clase navegación u otro menú. Este es modelado por un objeto compuesto que contiene un número fijado de ítems de menú. Cada ítem de menú tiene un nombre constante y posee un enlace, ya sea a una instancia de una clase de navegación o a un elemento de acceso. Cualquier menú es una instancia de alguna clase menú que es estereotipada por <<*menú*>> con su icono correspondiente. La propiedad *{frozen}* es adjuntada a cada atributo en una clase de ítem menú para indicar que los ítems de menú tienen nombres fijos. No obstante, la misma clase de ítem menú puede tener instancias diferentes ya que puede haber ítems de menú con el mismo nombre pero enlazadas a objetos diferentes (Koch et al., 2001).

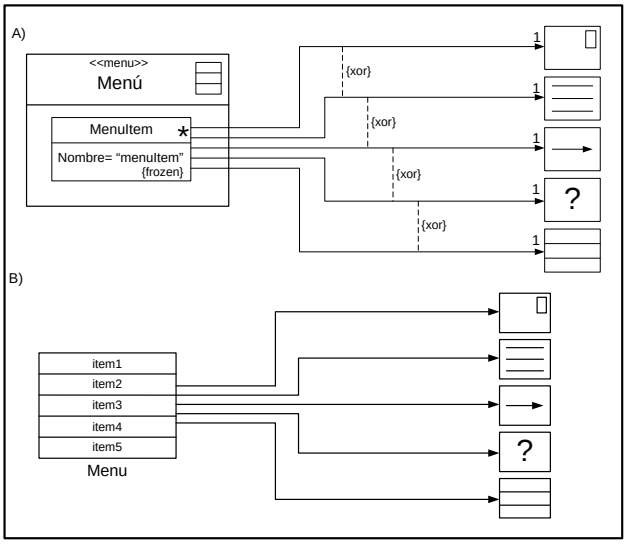

Figura 14: a) Clase Menú y b) su taquigrafía
Fuente: (Koch et al., 2001; Hennicker & Koch, 2001)

Las siguientes reglas son seguidas (Koch et al., 2001):

1. Considerar esas asociaciones que tienen como fuente una clase navegación.

2. Asociar a cada clase navegación, que tiene en el modelo previo al menos una asociación de salida, una clase menú correspondiente. La asociación entre una clase navegación y su correspondiente clase menú es una composición.

3. Reorganizar un menú en un menú con submenús.

4. Introducir por cada role, que ocurre en el modelo previo al lado final de una asociación dirigida un menú ítem correspondiente. Por defecto, el nombre del rol es usado como nombre constante del menú ítem.

5. Cualquier asociación del modelo previo que tiene como origen a una clase navegación ahora se convierte en una asociación del menú ítem correspondiente introducido en el paso 4. Notar que todos los pasos en el método anterior pueden ser realizados en una forma automática. Como resultado se obtiene un modelo de estructura de navegación comprensible de la aplicación.

5.4. Modelo de Presentación

El diseño de presentación soporta la construcción de un modelo de presentación basado en el modelo de estructura de navegación e información adicional, se recolecta durante el análisis de requerimientos. El modelo de presentación consiste en un conjunto de vistas que muestran el contenido y la estructura de los nodos simples, es decir cómo cada nodo es presentado al usuario y cómo el usuario puede

interactuar con ellos. Se propone la construcción de *sketches*[8], *storyboards*[9] y modelos de flujo de presentación (Koch et al., 2001).

Primero, el diseñador propone un sketch de cada vista de interfase de usuario principal, es decir el diseño de la interfase abstracta de usuario. Estos son dibujos a mano alzada de un par de elementos relevantes de cada nodo de navegación. Esta técnica de bosquejo es frecuentemente usada por diseñadores Web, pero sin tener una notación precisa para ello. Se propone usar una extensión de UML par este propósito. Estos sketches son usados para el modelo de storyboard que consiste en bosquejar escenarios. En el segundo paso basado en el modelo de storyboard el diseñador puede decidir si quiere escoger una técnica de *ventanas múltiples* o si quiere utilizar *frames*. El objetivo del modelo de flujo de presentación es mostrar donde las vistas de interfase de usuario del modelo de storyboard son presentadas al usuario, es decir en qué frame o ventana estas son desplegadas. Esto también muestra qué contenido es reemplazado cuando un usuario interactúa con el sistema (Koch et al., 2001).

5.4.1. Modelo de Interfase de Usuario Abstracta

El diseño de storyboard puede ser considerado un paso tan opcional como el diseño relacionado a la interfase de usuario. Los sketches dan un primer *look and feel* de la interfase. Después de haber producido las diferentes vistas de interfases de usuario (sketches) los escenarios de la realización del storyboard pueden ser desarrollados, los cuales muestran secuencias de las vistas en el orden en el cual el usuario puede navegar de una vista a otra. El objetivo es visualizar la organización de la estructura de la aplicación Web de una manera más intuitiva que en la fase de modelado de la estructura de navegación. Ambos, los sketches de las vistas así como la realización de los storyboards de los escenario, son medios muy útiles para la comunicación entre un cliente y el diseñador Web (Koch et al., 2001).

Para la construcción de los sketches se tiene un conjunto de elementos de modelado, tal y como se muestra en la figura 15 (Koch et al., 2001):

Las *vistas de interfase de usuario*. Una *interfase de usuario* (UI) especifica que cada instancia de esta clase es un contenedor de todos los elementos abstractos de interfases de usuario los cuales están presentados simultáneamente (es decir en un momento en una ventana) al usuario. Para las vistas de interfase de usuario se utiliza el estereotipo <<*UI view*>> y su respectivo icono.

La *clase presentación* es una unidad estructural que permite particionar una vista de interfase de usuario dentro de grupos de elementos de interfase de usuario. Para la clase presentación se utiliza el estereotipo <<*presentation class*>>.

[8] Se trata de un anglicismo conocido, cuya traducción al español es "bosquejo", es por esta razón que se utilizará la expresión en su idioma original

[9] Se trata de un anglicismo conocido, cuya traducción al español es "guiones gráficos", es por esta razón que se utilizará la expresión en su idioma original

El *elemento de interfase de usuario* es una clase de abstracción que tiene varias especializaciones describiendo elementos de interfase particulares.

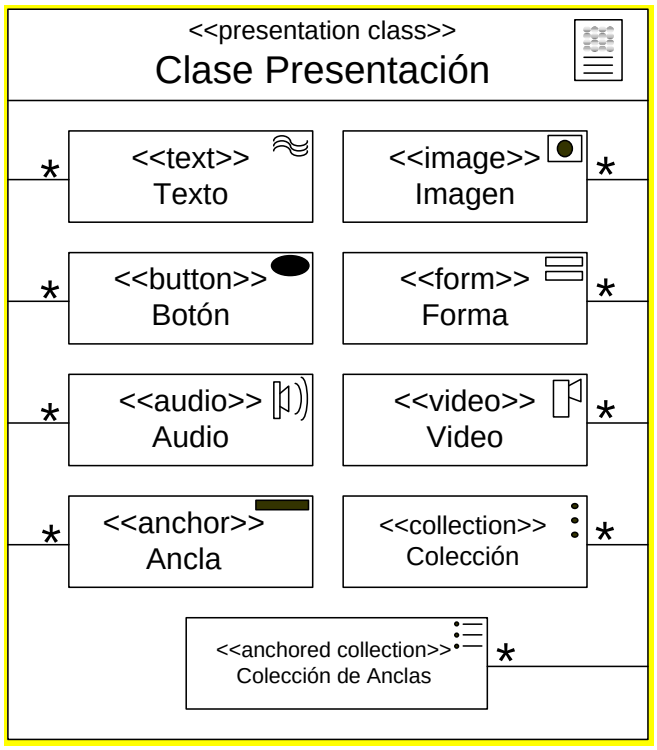

Figura 15: Clase Presentación como contenedor de otros elementos del Modelo de Presentación
Fuente: (Koch., 2000)

Para diseñar el modelo de storyboard se empieza con el modelo de navegación de la aplicación Web. Cada interfase de usuario abstracta es representada como una composición de clases. Las siguientes reglas pueden ser usadas para construir el modelo de presentación basado en las vista de interfase de usuario (Koch et al., 2001):

1. Construir una clase de presentación para cada clase de navegación que ocurra en el modelo de estructura de navegación. La clase presentación define una plantilla apropiada para presentar las instancias de la clase tomando en cuenta los atributos dados. Los elementos de interfase estereotipados tales como <<*text*>>, <<*image*>>, <<*audio*>>, <<*video*>>son usados por atributos de tipos primitivos y <<*collections*>>es usado para listas.

2. Construir una clase presentación para cada menú e índice que ocurra en el modelo de estructura de navegación. Esta presentación normalmente consiste de una lista de anclas. Los estereotipos <<*anchor*>> o <<*anchored collection*>>son usados para este propósito.

3. Construir una clase presentación para cada pregunta y visita guiada. Para las preguntas usar un estereotipo <<*form*>> y para las visitas guiadas usar un menú con los ítems *next* y *previous* (permite navegar al siguiente y al objeto previo sin una visita guiada)

4. Construir clases de presentación para soportar la navegación como composición de clases de presentación derivadas de las estructuras de acceso. Son usadas para reflejar el camino de navegación. Es decisión del diseñador donde incluir estas clases de presentación derivadas.

5. Añadir anclas a las clases de presentación para permitir la creación, destrucción y ejecución de operaciones sobre objetos del modelo conceptual. Los requerimientos funcionales de estas anclas provienen del modelo de caso de uso.

6. Determinar qué elementos de presentación deberían ser presentadas juntas al usuario (en una sola ventana). Las clases de presentación correspondientes deben ser compuestas en la vista de interfase de usuario (estereotipadas por <<*UI view*>>). Ya que el usuario necesita siempre una combinación de datos conceptuales y facilidades en la navegación, típicamente una vista de interfase de usuario consiste de la clase de presentación construida para la clase de navegación y de una clase de presentación construida para facilitar la navegación.

7. Construir escenarios de storyboard representadas por secuencias de vistas de interfases de usuario (opcional). Para este propósito se introducen enlaces que conectan un ancla (dentro de una vista UI) con otra vista UI para mostrar de esta manera los posibles flujos de presentación que puede ser causado por interacción de usuarios

5.4.2. Modelo de Estructura y Flujo de Presentación

El enfoque de este paso es modelar las dinámicas de la presentación mostrando dónde los objetos de navegación y los elementos de acceso serán presentadas al usuario, es decir en qué frames o ventanas el contenido es desplegado y qué contenido será reemplazado cuando un enlace es activado. Primero que nada, el diseñador tiene que especificar si es que una sola técnica es usada, si es que los frames son usados, y si es así, dentro de cuantos está dividido. En el caso de una ventana sin frames el resultado es obviamente proviene del modelo de storyboard y no es necesaria una representación gráfica (Koch et al., 2001).

Un modelo de flujo de presentación de una aplicación Web es construido con clases estereotipadas tales como <<*windows*>>, <<*frameset*>> y <<*frame*>>. Se usan estos estereotipos para indicar la locación de la presentación (Koch et al., 2001).

La *ventana (window)* (ver figura 16) es el área de la interfase de usuario donde los objetos de presentación son desplegados. Una ventana puede ser movida,

maximizada, minimizada, cambiada de tamaño, reducida, reducida a un icono y cerrada. Para realizar estas acciones una ventana contiene botones especiales (Koch et al., 2001).

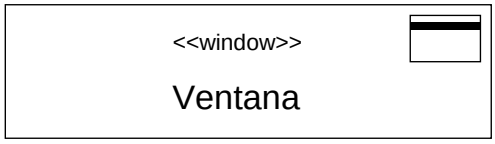

Figura 16: Ventana
Fuente: (Koch et al., 2001)

Un *frameset* (ver figura 17) es un elemento de modelado usado para definir áreas de visualización múltiple dentro de la ventana. Está dividida entre niveles bajos de localización de elementos y pueden contener también un número arbitrario de framesets anidados. Un frameset es una instancia de la clase frameset y está estereotipada por <<*frameset*>>con un icono correspondiente (Koch et al., 2001).

Un *frame* (ver figura 17) es siempre parte de un frameset, define un área del correspondiente frameset donde el contenido es desplegado. Un frame es una instancia de una clase frame estereotipada por <<*frame*>> con un icono correspondiente (Koch et al., 2001).

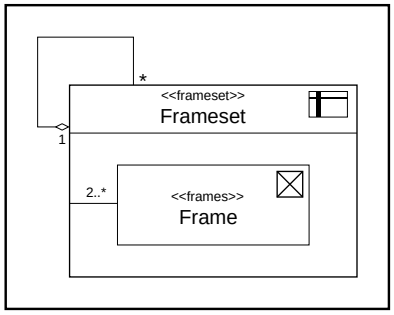

Figura 17: FrameSet y Frame
Fuente: (Koch et al., 2001)

El modelo de presentación requiere que el diseñador tome algunas decisiones tales como el número de ventanas a ser usadas y el número de frames en la que está dividida. Por lo tanto, la construcción de la estructura de presentación no puede ser automatizada por completo, pero existen ciertas guías que el diseñador debe seguir (Koch et al., 2001):

1. Escoger entre una técnica para ver una ventana o múltiples ventanas. En caso de elegir una técnica para ver múltiples ventanas planificar cómo muchas ventanas serán usadas.

2. Escoger el estilo del frameset, es decir con o sin frames. En el primer caso especificar cuántos frames tiene cada frameset.

3. Representar la estructura de presentación con un diagrama de clases UML (opcional).

4. Establecer el escenario para el modelo de interacción, o sea definir cuál camino de navegación del diagrama de estructura de navegación será modelado. Un camino de navegación relacionada a una caso de uso.

5. Representar a los usuarios, ventanas y los objetos frame en la dimensión horizontal.

6. Especificar un mensaje de *despliegue* para cada objeto de presentación que debería ser presentada al usuario (en una ventana o frame). El parámetro del mensaje de despliegue es el objeto de presentación correspondiente.

7. Incluir un mensaje de *selección* para cada acción de usuario que selecciona un ancla o un botón. Los nombres de anclas o botones son los parámetros del mensaje.

8. Especificar un mensaje de *llenado* y un mensaje de *introducción* para cada acción de usuario, que consiste en suministrar datos en una forma con preguntas. Esta forma es el parámetro de los mensajes.

9. Incluir un mensaje por cada apertura y cada cierre de una ventana.

6. Comentario

La aplicación de la IWeb sobre proyectos de esta naturaleza permite la construcción de aplicaciones Web funcionales y de calidad, y que además puedan mantenerse más fácilmente. De todo ello se ha hablado con anterioridad, de las metodologías, de los procesos y procedimientos, de las herramientas y técnicas, de las destrezas y habilidades, que permiten llevar a cabo la tarea de desarrollo de aplicaciones y sistemas basados en Web con más éxito. Sin embargo, existe una situación que se debe discutir a nivel local ¿Es posible aplicar todas estas guías y metodologías de la Ingeniería Web a la práctica de desarrollar aplicaciones Web? En países desarrollados este cuestionamiento es más una afirmación, prueba de ello es que la mayoría de los autores citados, que pertenecen a estos países desarrollados, basan sus investigaciones en sus experiencias propias en el campo laboral en el que se desenvuelven. En un país como Bolivia es posible que esto también suceda, pero para responder a la pregunta sin poseer datos precisos valdría la pena analizar la situación del medio profesional, en el que a su vez los aspectos que intervienen son de tipo social, cultural y económico.

En tal medio se suelen manejar mitos y creencias, como en muchos otros, que afectan de muchas formas a la aplicación de la Ingeniería del Software y de la Ingeniería Web. De hecho la palabra "Ingeniería" produce en algún caso cierta intimidación, ya que el mito que se genera con respecto a esta palabra es que sólo es posible de aplicarse a proyectos de gran envergadura; y que en el caso de proyectos

más pequeños esta aplicación más que una ayuda resulta un perjuicio porque extiende el tiempo y los costos de ejecución. A esto se suma la creencia de que el equipo de desarrolladores debe ser numeroso para que un proyecto se lleve a cabo y con éxito, pero lastimosamente los factores económicos impiden que se conformen equipos de trabajo grandes, así que los desarrolladores deben arreglárselas como puedan para concretar sus proyectos y no hay tiempo ni presupuesto para pensar en aplicar ingeniería.

Otro aspecto que influye enormemente es el comportamiento de la competencia y de los contratistas de proyectos. En el medio profesional del informático y del ingeniero de sistemas existe una práctica de competencia desleal. Imagínese la siguiente situación: algún desarrollador ofrece la cotización y el cronograma temporal de algún sistema que alguna entidad encarga para ser elaborada, pero al saber que le ofrecieron el proyecto a otra persona descubre que este otro desarrollador ofreció el mismo sistema a la mitad del precio y en menos tiempo. Es lógico darse cuenta porque no eligieron su propuesta. El segundo desarrollador ofrece el mismo sistema, pero acaso este se igualará en calidad, o le asegurará un buen rendimiento y ciclo de vida más largo al cliente. La duda se genera por las condiciones del trato puesto que es incierto si aplica o no métodos y procesos ingenieriles.

Lo desalentador del asunto es que las personas que suelen contratar estos trabajos tienen ese mito bien presente, para aclarar esto es necesario comentar que alguien alguna vez dijo textualmente: "Prefiero pagarle a una persona que desarrolle un sistema como sea en seis meses, a pagarle a otra persona que utilizará ingeniería del software y terminará el sistema en cinco años". Esa relación no es necesariamente cierta, es más puede que en la práctica suceda exactamente lo contrario, ya que no se está considerando como va a funcionar un sistema después de su entrega, y tampoco los costos de este en un largo plazo.

Además, en el medio profesional, al informático e ingeniero de sistemas no se les atribuyen funciones adecuadas según su conocimiento. La creencia de la mayoría de las personas es que el informático e ingeniero de sistemas son expertos en "todo" lo que a computadoras se refiere, y que es su responsabilidad el buen o mal funcionamiento tanto del hardware como del software. Esperar que eso sea real es como esperar que un médico sea experto en todas las ramas de la medicina, tales como la oftalmología, neurología, ginecología, y muchas otras; el conocimiento que puede tener de todas estas ramas es de tipo general, al igual que un informático y un ingeniero de sistemas tienen acerca de todas las ramas de la ciencia de la computación, pero el hecho de ser un experto en todas las áreas es demasiado para cualquier ser humano.

En cuanto al desarrollo Web, son diseñadores gráficos los que se ocupan de esta tarea, o en el mejor de los casos informáticos e ingenieros que aplican sus conocimientos en diseño gráfico. En algunas instituciones sí se emplean esfuerzos para construir aplicaciones y sistemas Web más complejos, sin embargo, lo más

probable es que las metodologías que se emplean sigan siendo ad hoc, y que la Ingeniería Web todavía no sea aplicada en el medio laboral.

La conclusión es que en el medio local, la ingeniería del software y la ingeniería Web se conocen, incluso son estudiadas, pero existe todavía una resistencia a su aplicación debido a los mitos y creencias mencionados antes. La esperanza es que es temprano todavía para afirmar cualquier cosa. Especialmente para la ingeniería Web, puesto que, su estudio, investigación y aplicación es reciente a nivel global. La recomendación es que se conforme otra realidad para el medio profesional, ya no forjada en los mitos y creencias, si no más bien, salida de la costumbre sana de recurrir a procesos y métodos disciplinados además de sólidas metodologías que la ingeniería del software y la ingeniería Web son capaces de brindar.

7. Bibliografía

Culebro, M., Gómez, W. & Torres, S. (2006). *Software libre vs. Software Propietario, Ventajas y Desventajas.* México: Creative Commons.

Deshpande, Y., Ginige, A., Murugesan, S., & Hansen, S. (2002). *Consolidating Web Engineering as a Discipline*. SEA Software, April 2002, (pp. 32-34).

Deshpande, Y., Chandrarathna, A., Ginige, A. (2002). *Web Site Auditing – First Step Towards Re-engineering.* 14th international conference on Software engineering and knowledge engineering SEKE '02. EE.

Escalona, J. & Koch, N. (2002). *Ingeniería de Requisitos en Aplicaciones para la Web -Un Estudio Comparativo*. Sevilla, España: Universidad de Sevilla.

Fraternali, P. (1999). *Web Development Tools: a Survey.* Agosto, 9, 2006. Milán, Italia: Dipartimento di Elettronica e Informazione.
http://www7.scu.edu.au/1835/com1835.htm.

Froufe, A. (2002). *JavaServer Pages. Manual de Usuario.* México D.F: ALFA OMEGA Grupo Editor, S.A. de C.V.

Galli, R. (2001). *Desarrollo Web Extremo.* España, Mallorca: Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores.

Ginige, A., Murugesan, S. (2001). *Web Engineering: An Introduction.* Australia: IEEE MultiMedia.

Hennicker, R., Koch, N. (2001). *A UML-based Methodology for Hypermedia Desing*. Institute of Computer Science Ludwig-Maximilians University of Munich. Munchen Germany.

Internet World Stats. (2006). *Internet Growth Statistics.* Mayo, 31, 2006. http://www.internetworldstats.com/emarketing.htm.

Koch, N., (2000). *Software Engineering for Adaptive Hypermedia Systems. Reference Model, Modeling Techniques and Development Process.* Ludwig-Maximilians-Universität München, Germany.

Koch, N., Kraus, A., Hennicker, R. (2001). *The Authoring Process of the UML-based Web Engineering Approach*. Institute of Computer Science Ludwig-Maximilians University of Munich. Munchen Germany.

Koch, N., Kraus, A., (2002). *The Expressive Power of UML-based Web Engineering.* Ludwig-Maximilians University of Munich. Munchen Germany

Manchón, E. (2003). *Errores comunes en el diseño de sitios Web.* Junio, 5, 2006. Creative Commons. http://www.alzado.org/articulo.php?id_art=49.

Merlo, J. (2003). *La Evaluación de la Calidad de la Información Web: Aportaciones Teóricas y Experiencias Prácticas*. Universidad de Salamanca.

Murugesan, S., Deshpande, Y.Hansen, S., Ginige, A. (1999). *Web Engineering: A New Discipline for Development of Web based Systems.* Proc. 1st ICSE Workshop on Web Engineering, Los Angeles, May 1999 (pp. 1-9).

Murugesan, S., Ginige, A. (2005). *Web Engineering: Introduction and Perspectives.* En Woojong Suh (Ed.), Web Engineering: Principles and Techniques (pp. 1-30). United States of America: Idea Group Publishing.

Olsina, L. (1999). *Ingeniería de Software en la Web. Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de sitios Web.* La Plata: Facultad de ciencias Exactas, Universidad Nacional de La Plata – Argentina.

Prats, J., Rovira, J. (eds.) (1992). *Diccionario Enciclopédico Ilustrado OCEANO UNO*. Bogotá: Ediciones Océano Gallach, S.A.

Pressman, R. (1998). *Can Internet-based Applications Be Engineered*? IEEE Software, September/October IEEE Press.

Pressman, R. (2002). *Ingeniería del Software. Un Enfoque Práctico (5ta ed.).* Madrid: McGRAM-HILL/ INTERAMERICANA DE ESPAÑA, S.A.U.

Ricca, F., Tonella, P. (2005). *Program Transformations for Web Application Restructuring.* En Woojong Suh (Ed.), Web Engineering: Principles and Techniques (pp. 242-260). United States of America: Idea Group Publishing.

Ruiz, F., Polo, M. (2001). *Mantenimiento de Software*. Grupo Alarcos, Dep. de Informática, Escuela Superior de Informática, Universidad De Castilla-La Mancha.

Spector, D. (2002). *CASE Tools: Large System Development.* Agosto, 9, 2006. Linux in the Enterprise.
http://www.linuxdevcenter.com/pub/a/linux/2002/08/01/enterprise.html.

Zuazo, A., (2004). *Selección de Auxiliares de Docencia Vía Web.* Carrera de Informática, Universidad Mayor de San Andrés. La Paz, Bolivia.