CENTRO DE INVESTIGACIÓN
Y DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE COMPUTACIÓN

TÍTULO

METODOLOGÍA PARA EL DISEÑO DE SISTEMAS DE INFORMACIÓN INTRANET

TESIS PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS
EN LA ESPECIALIDAD DE INGENIERÍA ELÉCTRICA

QUE PRESENTA:
ING. ULISES JUÁREZ MARTÍNEZ

DIRECTOR DE TESIS:
DR. SERGIO VÍCTOR CHAPA VERGARA

MÉXICO, D.F., MAYO DE 1999


XM
CLASIF.
ADDSUB.
FECHA
PROCED.
$

A mi Madre

María Emma

Agradecimientos

A mi madre, por todo su apoyo en mis estudios, por la confianza que siempre me ha tenido y por soportar mi carácter.

A mi abuelo, a mis tíos y en memoria de mi abuela, por el apoyo brindado, en especial los últimos dos años.

A mi padre, por la ayuda en la adquisición de bibliografía.

Al Dr. Sergio Chapa por su asesoría y amistad.

A la Sra. Sofía Reza Cruz por su apoyo secretarial.

Al CONACyT por el apoyo de la beca de manutención otorgada durante el plan de estudios.

A Aurelio Hernández Ramírez, por esa gran amistad y apoyo que me ha dado, aún estando al otro lado del mundo.

A Maricruz Balderas, Enrique Martínez y Nicolás Rodríguez.

Resumen

Los Sistemas de Información Intranet incorporan una serie de características diversas que comprenden principalmente: diversos dialectos derivados del SGML (HTML, XML), generación dinámica de páginas, lenguajes de consulta (SQL), lenguajes de programación (3GL y 4GL) y arquitecturas de componentes (CORBA). La etapa del diseño para estos sistemas requiere un soporte adecuado para integrar de forma correcta la naturaleza de la Web. Trabajos desarrollados por diversos autores no contemplan todos los factores, sólo se centran en los aspectos de navegación, descomposición jerárquica de una página web y enfoques relacionales.

El objetivo del presente trabajo ha sido desarrollar una metodología para el diseño de Sistemas de Información Intranet. Enfoques evolutivos e iterativos de ingeniería del software, así como modelos de análisis y diseño tradicionales u orientados a objetos pueden aplicarse a los Sistemas de Información Intranet. Para complementar el diseño, es necesario que en la etapa de análisis se enfatice sobre la obtención de interfaces primarias.

La participación en proyectos de desarrollo de sistemas de información intranet ha permitido retroalimentar la propuesta de diseño y al mismo tiempo, la metodología mantiene las cuatro componentes fundamentales del diseño: diseño de arquitectura, diseño de datos, diseño de interfaz de usuario y diseño de procesos.

Los Diagramas de Hipervínculos obtenidos mostraron ser una herramienta que no solo representa la arquitectura del sistema, también permite representar rutas de navegación y secuencias de interfaces, principalmente. Se han incorporado prototipos conceptuales para el correcto entendimiento del dominio del problema, la ubicación correcta de los asuntos del negocio, definición de interfaces y su funcionalidad.

Para lograr la integración de todos los elementos Web que pueden participar en la construcción de un sistema, se ha desarrollado el concepto de Interacción de Lógicas, con el cual se definen ámbitos que cumplen una función específica y pueden corresponder a algún elemento Web. El Lenguaje de Definición de Procesos Intranet facilita el diseño de los procesos al aislar funciones específicas dentro de una entidad web o de un proceso y eliminar detalles de codificación.

De ésta forma, los enfoques estructurado y orientado a objetos son complementados con la metodología de diseño propuesta al permitir la incorporación de la naturaleza Web en el desarrollo de sistemas. La aplicación de la metodología ha mostrado resultados exitosos tanto en Sistemas de Información Intranet, como en páginas Web tradicionales.

Palabras clave: Ingeniería del software, Sistemas de información, Web, Internet, Intranet, HTML, XML, Cliente/Servidor, CORBA, UML.

Contenido

Relación de figuras

Relación de tablas

Capítulo 1. Introducción

1.1. Objetivos

El objetivo general del presente trabajo es la propuesta de una metodología para el diseño de Sistemas de Información Intranet (SII), que incluye:

- Desarrollar una metodología para el diseño de SII.
- Determinar cuáles son los aspectos más convenientes de las metodologías de diseño estructurado y orientado a objetos, que aporten elementos para el diseño de SII.
- Determinar a partir de desarrollos de sistemas reales, aquellos aspectos favorables que proporcionan las tecnologías Web y herramientas de desarrollo, que permitan una mejor construcción y optimización de los SII.

1.2. Antecedentes

Se han desarrollado una serie de trabajos que se centran en el diseño y construcción de entornos web: algunos tienen aportaciones interesantes dentro la ingeniería de software, otros sólo se limitan a los aspectos estáticos. El interés de los autores no sólo es el disponer de un método para diseñar sistemas web, sino también de una herramienta que soporte al método.

Elementos como: texto de navegación, navegación local, botones de ruta e iconos de navegación son los que comprenden el diseño genérico de la navegación. Bachiochi y colaboradores [hBac] han realizado estudios de usabilidad para el diseño navegacional recomendando botones con etiquetas apropiadas y arreglos en la parte superior de la página, para ofrecer una estructura de botones asociada a las páginas web. De acuerdo a la disposición de los arreglos de los botones en la parte superior, el resultado es la disminución del tiempo empleado en la navegación de las páginas.

El **Modelo de Composición Web** se basa en un principio de descomposición jerárquica de una página web, en donde, en el nivel más alto de la jerarquía aparecen las páginas y diálogos que pueden establecerse por un número de páginas interrelacionadas, y en el nivel bajo de la jerarquía se tienen las tablas, anclas y referencias a fuentes dinámicas. Otros componentes en la página se llaman compósitos. Cada uno de los componentes puede ser

un prototipo que describe propiedades de un grupo de componentes. Esta descomposición permite acceso a los conceptos de diseño para desarrollar abstracciones de alto nivel como estructuras de ligas, y la reutilización de componentes [hGel].

Una variante al Modelo de Composición Web es la presentada por los W3Objects [hIng], la cual mantiene la noción de grano grueso de objetos que encapsulan fuentes, como lo sería la integración de la Web con objetos distribuidos. Destaca el manejo de aplicaciones basado en sesiones.

Otros trabajos han empleado enfoques relacionales, los cuales se han utilizado para describir los aspectos estáticos de las páginas web. También incorporan el desarrollo de escenarios para poder establecer y manejar los aspectos dinámicos, como son el acceso, uso e intercambio entre los usuarios y el sistema web [hTak]. Los atributos definidos para cada una de las páginas web permiten realizar el mantenimiento de las mismas. Los enfoques relacionales y el uso de escenarios definen etapas de análisis y diseño, las cuales permiten trabajar de forma más cercana a los procesos de ingeniería de software.

1.3. Internet

Internet es una red de redes, que conecta universidades, escuelas, empresas y gente en una supercarretera de información nunca antes imaginada. Cuenta con una infraestructura de líneas telefónicas digitales de alta velocidad con transmisiones de más de 40 megabytes por segundo. Operada inicialmente por la National Science Fundation (NSF), Internet integró principalmente a instituciones gubernamentales, educativas, y de investigación; y en 1994, cede la operación de Internet a los Proveedores de Servicios Internet (PSI).

La World Wide Web (WWW, W3 o simplemente la Web), es una armazón arquitectónica para acceder a documentos vinculados y distribuidos en miles de máquinas en toda la Internet. En cinco años pasó de ser tan sólo una manera de distribuir datos sobre experimentos de física de altas energías, a la aplicación con una interfaz gráfica que es fácil de usar por los principiantes, proporcionando de esta forma el acceso a casi cualquier tema imaginable disponible en Internet [Tan97].

Los PSI y el surgimiento de la WWW sobre Internet facilitó su apertura hacia el uso comercial. La necesidad de proteger datos no se hizo esperar y la incorporación de *firewalls* (muros de protección/seguridad) permitió a muchas compañías obtener privacidad. El esquema público de Internet empezó a cambiar por ciertos sitios protegidos, en los cuales sólo se tenía acceso desde la propia compañía.

1.4. Intranets

Las Intranets tienen poco tiempo. Se dice que se originaron al describir el desarrollo de una solución interna de cliente/servidor basada en tecnología Web. El Dr. Steve Telleen at Amdahl usó el término IntraNet al inicio de 1994 en un artículo que escribió sobre

metodología IntraNet ubicada en un sitio privado, colocada posteriormente en Internet. La primera impresión comercial se encontró en el artículo de Stephen Lawton [Law95], donde se discute que cerca de 1000 compañías colocaban páginas e instalaban servidores telnet y ftp.

Los pioneros fueron Boeing, Schlumberger Ltd., Weyerhaeuser Corp., Sun Microsystems, y Digital Equipment Corp. Las ventajas fueron listadas como bajo costo, fácil de escribir en HTML (Hypertext Markup Language), y acceso a varios paquetes de documentos en línea, tales como manuales de empleado, material de investigación y páginas base individuales [Hin97].

Una Intranet se define como una red dentro de una organización (red interna) que adapta tecnologías Internet para usarlas en su infraestructura de información [Abl96]. Esta definición es una de las más generales; sin embargo, el concepto de una Intranet se puede extender hacia dos puntos de vista [Hin97]:

- Definición técnica. Una Intranet es un ambiente computacional heterogéneo que conecta diferentes plataformas de hardware, ambientes de sistemas operativos, e interfaces de usuario para: comunicar, colaborar, efectuar transacciones e innovar.
- Definición organizacional. Una Intranet es una organización que es capaz de integrar gente, procesos, procedimientos, y principios para formar una cultura creativa intelectualmente dedicada a la implementación total de la efectividad organizacional.

Un valor agregado y anticipado es la Extranet, la cual permite dirigir negocios en línea con otras Intranets de compañías directamente sobre la Internet. Las Extranets son Intranets de servicio completo que combinan todos los servicios de la red con todos los servicios tecnológicos de la Web, que incluyen a Internet y la WWW.

1.5. Sistemas de Información Intranet (SII)

El uso de protocolos de internet (IP: Internet Protocol) en redes LAN (Local Area Network) ha desplazado a otros protocolos, como IPX y AppleTalk, y los sistemas de información desarrollados sobre éstos últimos, se han convertido a sistemas de información que hacen uso de la tecnología Web. De ésta forma, se dispone ahora de Sistemas de Información Intranet (SII), los cuales son sistemas de información cliente/servidor basados en tecnologías de Internet, con conexión opcional a la misma.

A medida que el almacenamiento de datos permita incorporar nuevos formatos y mejorar las capacidades actuales disponibles, se tendrá mayor diversidad de elementos que serán necesarios incorporar en los sistemas de información. Los principales elementos a considerar son los siguientes [Orf96]:

- **Lenguajes de consulta.** Las especificaciones de dichos lenguajes son una pieza clave para las capacidades de los sistemas en general, como lo es SQL3.
- **Bodegas de datos.** Son depósitos de datos que mediante un procesamiento adecuado sirven para ayudar en la toma de decisiones. Se caracterizan por contener información seleccionada o filtrada, la cual proviene de un gran número de bases de datos de

producción. Las bodegas pueden entonces jerarquizarse y obtener funcionalidad específica, a lo cual se le llama Datamarts.

- **Datos multidimensionales.** Los datos multidimensionales se pueden obtener de una base relacional o una base especializada. La base especializada es una DBMS multidimensional (MDBMS) la cual proporciona mecanismos de bases de datos especializados para almacenar datos en matrices a lo largo de dimensiones relacionadas llamadas hipercubos. Emplean esquemas de indexación intensiva para optimizar el acceso a estos cubos.
- **Minería de datos.** La minería de datos son mecanismos que permiten clasificar enormes cantidades de información para la determinación de patrones valiosos en los datos.
- **Componentes.** Son recursos de software reutilizables, los cuales se almacenan y clasifican en una biblioteca o repositorio.

1.5.1. Enfoque de diseño para los SII

Las tendencias en los SII, tanto en la aparición de nuevas tecnologías de la Web como en la evolución de otras ya existentes, definen aspectos que se deben considerar dentro de un desarrollo de SII. Tales aspectos comprenden:

- La incorporación de lenguajes derivados del SGML (Standar Generalized Markup Language).
- La dinámica e interacción ofrecida por el lenguaje Java.
- Los aspectos de generación dinámica de páginas.
- El manejo de scripts para la funcionalidad de la interfaz en HTML.
- Las capacidades de reutilización por medio de componentes.
- Las interacciones entre diferentes lenguajes de programación.

Los puntos anteriores son necesarios considerarlos para el diseño y construcción de SII. Las metodologías empleadas en la ingeniería de software requieren ser complementadas con métodos que consideren y soporten los aspectos citados, los cuales son característicos de un ambiente web.

Capítulo 2. Ingeniería del Software Cliente/Servidor

Los sistemas cliente/servidor han evolucionado junto con los avances de la computación personal, con nuevas tecnologías de almacenamiento, mejores comunicaciones por red y mejor tecnología de bases de datos. Los métodos tradicionales y orientados a objetos empleados en ingeniería de software se utilizan en los sistemas cliente/servidor.

El objetivo de este capítulo es revisar brevemente la estructura de los sistemas cliente/servidor, así como los métodos de análisis y diseño que se han considerado para dar soporte a los mismos.

2.1. Estructura de los sistemas cliente/servidor

De forma general, una arquitectura distribuida y cooperativa tiene un sistema raíz, el cual es una computadora de gran capacidad que almacena datos corporativos. El sistema raíz conecta a servidores, los cuales son estaciones de trabajo que realizan solicitudes de datos corporativos. Cada servidor mantiene un sistema departamental a través de una LAN. La Figura 1 muestra la arquitectura distribuida.

Las diferentes implementaciones que se pueden realizar dentro de una arquitectura cliente/servidor son las siguientes:

- **Servidores de archivos.** El cliente solicita registros específicos de un archivo y el servidor transmite estos registros al cliente a través de la red.
- **Servidores de bases de datos.** El cliente envía solicitudes de SQL en forma de mensajes al servidor. El servidor procesa las solicitudes SQL, recupera los datos y los envía al cliente.
- **Servidores de transacciones.** El cliente envía una solicitud que invoca procedimientos remotos (por ejemplo, un grupo de sentencias SQL). Se efectúa la transacción y el resultado es enviado al cliente.
- **Servidores de grupos de trabajo (groupware).** La comunicación entre clientes (y la personas que los usan) se puede realizar por medio de texto, imágenes, correo, tableros electrónicos y otras presentaciones. Cuando se tienen estos servicios se tiene una arquitectura de grupos de trabajo.

- **Servidores de objetos.** Los objetos del cliente se comunican con los objetos del servidor mediante un corredor de solicitudes de objetos (ORB: *Object Request Broker*). El cliente invoca un método de un objeto remoto. El ORB localiza una instancia de esa clase de servidor de objetos, invoca el método solicitado y envía los resultados al objeto cliente.
- **Servidores web.** Los clientes y los servidores se comunican mediante un protocolo semejante a RPC llamado HTTP. Este protocolo define un conjunto simple de órdenes, los parámetros se transmiten en cadenas, sin estipulaciones de datos tecleados.

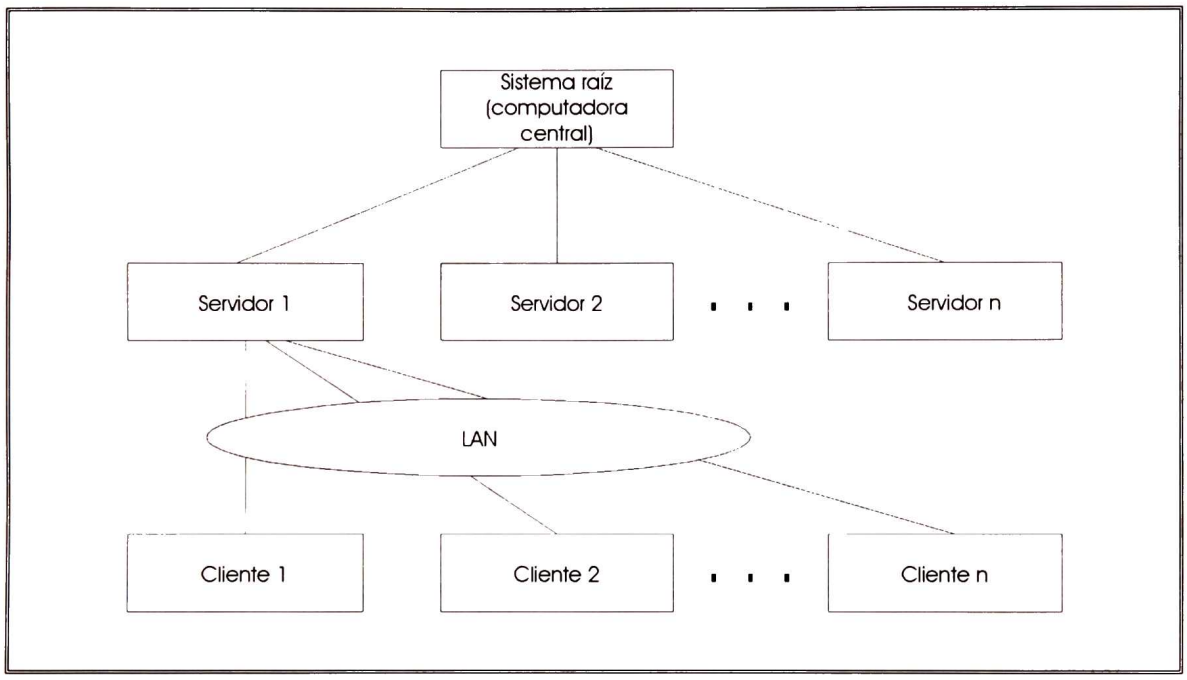


Figura 1. Arquitectura distribuida en un entorno corporativo.

2.1.1. Componentes de software para sistemas C/S

La arquitectura de los sistemas C/S requiere componentes de software que se asocian al cliente, al servidor o a ambos.

- Los componentes de interacción y presentación (capa de presentación). Implementa las funciones que normalmente se asocian a una interfaz de usuario.
- Componentes de aplicación. Implementa las reglas del negocio (capa lógica del negocio). Normalmente estos componentes pueden descomponerse para residir tanto en el cliente como en el servidor.

- Los componentes de gestión de bases de datos (capa de administración de datos). Llevan a cabo la manipulación y la gestión de datos requerida por una aplicación.

La siguiente figura esquematiza las capas que forman los componentes y la interacción que se realiza entre cada una de ellas.

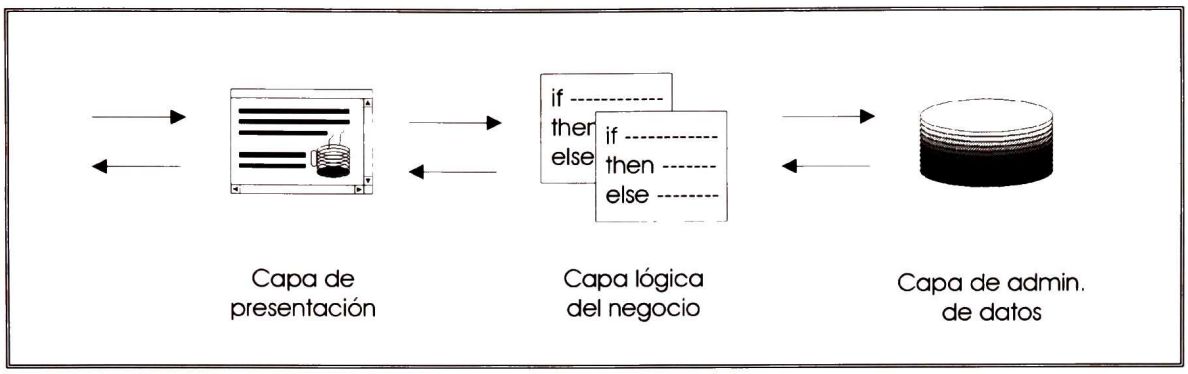


Figura 2. Capas de software.

Existe otro bloque de construcción de software, particularmente para el software de comunicación cliente/servidor, llamado *middleware* o software intermedio. El *middleware* empieza en el módulo de la API de la parte del cliente que se emplea para invocar un servicio y comprende la transmisión de la solicitud por red y la respuesta resultante.

2.1.2. Distribución de componentes de software

Cuando la mayor parte de la funcionalidad de los componentes se asocia al servidor, se tiene un servidor principal. De la misma manera, cuando el cliente implementa la mayor parte de los componentes se tiene un diseño de cliente principal.

Los clientes principales se obtienen cuando se implementan arquitecturas de servidor de archivos y de servidor de bases de datos. Los servidores principales se obtienen cuando se implementan sistemas de transacciones y de trabajo en grupo.

La ubicación de componentes también permite definir otro tipo de configuraciones. Cuando una aplicación tiene alguna configuración de cliente principal o servidor principal (e incluso ambas), se dice que la arquitectura está en dos planos. Es posible que la implementación de los componentes se encuentre totalmente independiente del servidor y del cliente; e incluso, puede estar separada de los datos y de la interfaz de usuario. Cuando se tiene esta característica se dice que la arquitectura está en tres planos.

Líneas generales para distribuir componentes

Actualmente no se tienen reglas definidas que indiquen o describan cómo distribuir componentes en el cliente, en el servidor o entre ambos. Sin embargo hay tres reglas generales que se han seguido para tal caso:

1. Los componentes de presentación/interacción suelen ubicarse en el cliente.
2. El acceso a datos por usuarios conectados por medio de una red, hace que la base de datos se ubique en el servidor.
3. Los datos estáticos que se utilicen como referencia deberán asignarse al cliente.

Los demás componentes deberán distribuirse entre el cliente y el servidor sobre la base de una distribución que optimice las configuraciones de cliente, servidor y red que los conecta. También con la evolución de las arquitecturas C/S, la tendencia es a ubicar la lógica de la aplicación volátil en el servidor, con lo cual se simplifica la actualización de las aplicaciones cuando se hacen cambios en la lógica de la aplicación.

2.1.3. Arquitectura común de corredores de solicitudes de objetos

Los componentes de software C/S están implementados mediante objetos que deben ser capaces de interactuar entre sí en una sola máquina o a través de la red. El ORB descrito en la sección anterior, permite interceptar el mensaje y maneja todas las actividades de comunicación y de coordinación que son necesarias para: localizar el objeto requerido, invocar el método con el paso de los datos adecuados, y transferir los datos resultantes al objeto que originó el mensaje.

El Object Management Group (OMG) ha desarrollado y publicado una arquitectura común de ORB llamada CORBA (*Common Object Request Broker Architecture*), la cual se ilustra en la siguiente figura.

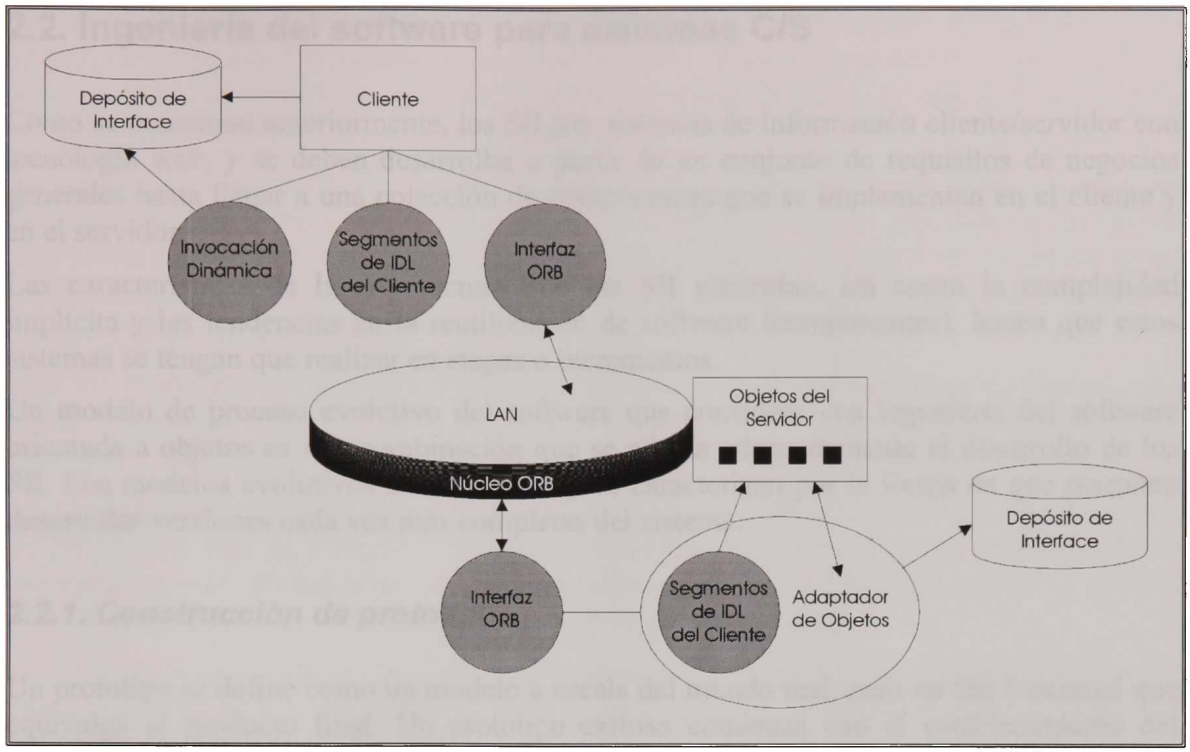


Figura 3 Arquitectura básica de CORBA.

Con CORBA, los objetos que residen tanto en el cliente como en el servidor se definen mediante el lenguaje de definición de interfaz (IDL: *Interface Definition Language*). El IDL es un lenguaje declarativo para definir objetos, sus métodos y atributos, los cuales son consistentes e independientes del lenguaje o de la implementación de la plataforma.

El depósito de interfaces es un mecanismo que permite almacenar las descripciones de los objetos y su ubicación, de tal forma que estén siempre disponibles para las solicitudes que se efectúan en el momento de la ejecución.

Cuando se realiza una invocación de un cliente a un método de un objeto servidor, se utiliza la invocación dinámica para:

1. Obtener la información necesaria del objeto a partir del depósito de interfaces.
2. Crear una estructura de datos con parámetros que se habrán de pasar al objeto.
3. Crear una solicitud para el objeto e invocarla.

La solicitud se pasa al núcleo ORB, la cual es procesada por el servidor en donde el adaptador de objetos guarda la información de clases y objetos en un depósito de interfaces del servidor, y posteriormente se obtiene la gestión de las solicitudes del cliente. Posteriormente se emplean otros segmentos del IDL como interfaz con la implementación real del objeto [Orf96] [Pre98].

2.2. Ingeniería del software para sistemas C/S

Como se mencionó anteriormente, los SII son sistemas de información cliente/servidor con tecnología web, y se deben desarrollar a partir de un conjunto de requisitos de negocios generales hasta llegar a una colección de componentes que se implementan en el cliente y en el servidor.

Las características de la arquitectura que los SII presentan, así como la complejidad implícita y las tendencias en la reutilización de software (componentes), hacen que estos sistemas se tengan que realizar en etapas o incrementos.

Un modelo de proceso evolutivo del software que considere una ingeniería del software orientada a objetos es una combinación que se adapta adecuadamente al desarrollo de los SII. Los modelos evolutivos son iterativos y se caracterizan por la forma en que permiten desarrollar versiones cada vez más completas del sistema.

2.2.1. Construcción de prototipos

Un prototipo se define como un modelo a escala del mundo real, pero no tan funcional que equivalga al producto final. Un prototipo exitoso comienza con el establecimiento del objetivo de lo que se quiere aprender al hacerlo. Una vez que se comprende lo que se quiere lograr, se puede seleccionar una técnica de creación de prototipos (de alta tecnología o de tecnología básica) que sea lo más costeable para lograr el objetivo [Rub97].

Aunque en las diferentes etapas de los proyectos un prototipo puede ser útil, lo más importante al respecto es que su objetivo debe ser siempre claro. Por ejemplo, durante la fase de análisis se puede usar un prototipo para obtener los requerimientos del usuario. En la etapa de diseño también puede ayudar a evaluar muchos aspectos de la implementación seleccionada.

Prototipos de tecnología básica

El prototipo más simple, es aquel que se construye con tecnología básica, tal como: un conjunto de pantallas en hojas de papel dispuestas en la pared, procesadores de texto, herramientas de dibujo y pizarrones en blanco. Todos los objetos, ventanas de diálogo y tipos de mensajes conocidos que integran una interfaz de usuario deben estar disponibles.

Las ventajas que presentan estos prototipos son el apoyo a actividades de análisis para obtener diagramas de entidad–relación, de transición de estados y asuntos del negocio, así como, obtener las disposiciones de ventanas y mapearlas al diagrama entidad–relación para obtener la complejidad de acceso a los datos.

La desventaja principal es que pueden emplearse mal, al considerar que los procesos de análisis deben estar terminados para su utilización.

Prototipos de alta tecnología

El prototipo también puede ser muy elaborado, construido con herramientas de software que permiten interacción con el usuario final a través del ratón, junto con la introducción de

datos. Estos prototipos son codificados desde su inicio y eventualmente evolucionan hacia un sistema terminado. El sistema se codifica en forma progresiva, al inicio se comienza con la disposición de la interfaz y posteriormente en forma iterativa, se agregan más y más detalles hasta satisfacer los objetivos planeados.

La garantía de la construcción de prototipos de alta tecnología, es definir desde el comienzo las reglas del juego, para que el usuario esté de acuerdo en que el prototipo se construya para servir sólo como un mecanismo de definición de los requisitos. Posteriormente ha de ser descartado (al menos en parte) y debe construirse el software real, para enfocarse en la calidad y en el mantenimiento.

Para los prototipos de alta tecnología, la pregunta no es si hay que construir un sistema piloto y tirarlo, se tirará. La única pregunta es si planificar de antemano la construcción de algo que se va a desechar, o prometer entregar el desecho a los usuarios finales [Bro75].

Las ventajas en estos prototipos son que permiten la reutilización de código (componentes ya probados), con lo cual se disminuye considerablemente el tiempo de desarrollo y permiten repartir el esfuerzo de programación entre varios programadores y realizar pruebas independientes de la aplicación por un equipo de calidad.

La desventaja principal surge cuando el usuario pide mejoras a un prototipo que no es funcional, y el gestor del proyecto llega a ceder porque se considera que el prototipo cumple con los objetivos.

2.2.2. El modelo en espiral

Este modelo tiene sus orígenes en el Pentágono, y se desarrolló como un método para tener control sobre los costos excesivos de armas masivas y proyectos de desarrollo de sistemas de defensa. En el caso de la industria del software, el modelo se ajustó ligeramente para las necesidades de desarrollo de sistemas grandes y complejos, y en la actualidad permite el desarrollo rápido de aplicaciones (RAD: *Rapid Application Development*).

El paradigma del modelo en espiral de la ingeniería de software, se desarrolló para incluir las mejores características de los paradigmas del ciclo de vida clásico y de la creación de prototipos, y con la incorporación de un nuevo elemento, el análisis del riesgo, el cual no está presente en esos paradigmas. El modelo en espiral define cuatro actividades principales [Pre93]:

1. Planificación. Determinación de objetivos, alternativas y restricciones.
2. Análisis de riesgo. Análisis de alternativas e identificación/resolución de riesgos.
3. Ingeniería. Desarrollo del producto a su nivel correspondiente.
4. Evaluación por el usuario final. Valoración de los resultados de la actividad de ingeniería.

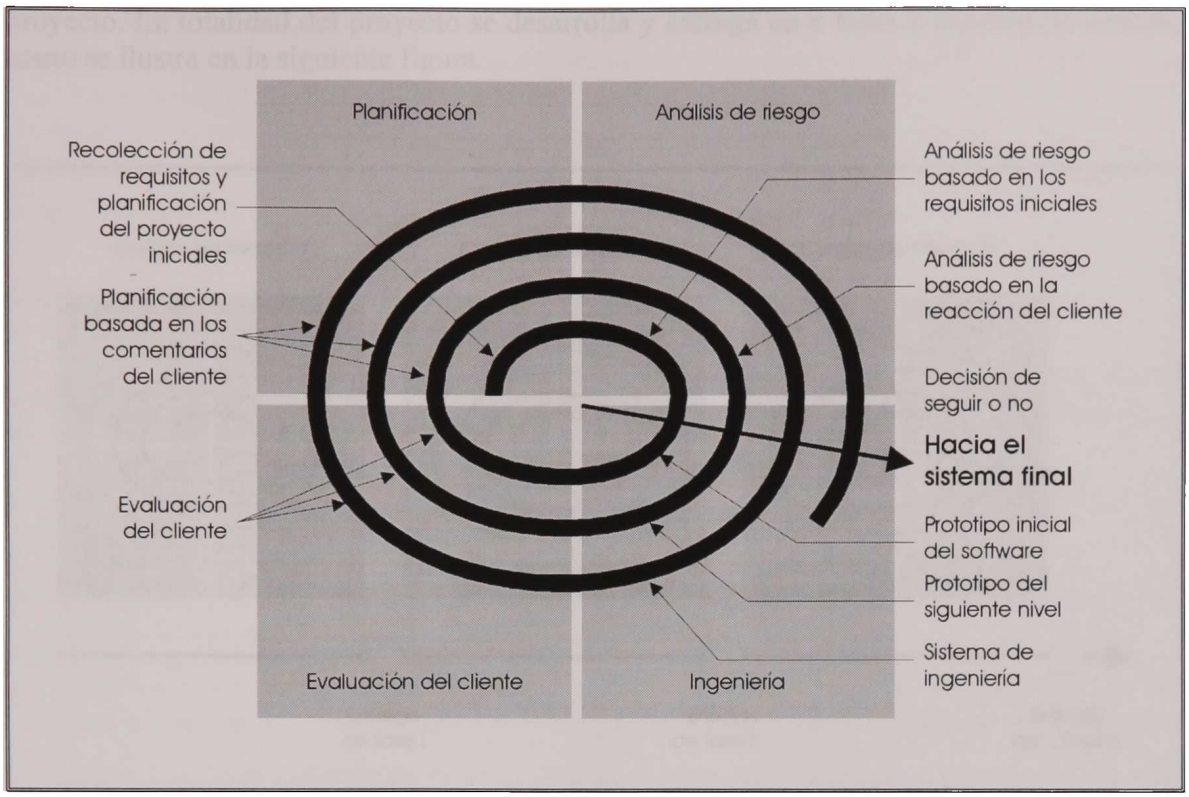


Figura 4. El modelo en espiral.

En cada ciclo del modelo, se desarrollan versiones sucesivas del software cada vez más completas. En la primera iteración se definen los objetivos, las alternativas y las restricciones, así como se analizan e identifican los riesgos, y en caso de ser necesario se puede hacer la integración de prototipos.

Al término de cada ciclo, el usuario final evalúa el trabajo resultante de la actividad de ingeniería, y sobre la base de sus críticas se pasa al siguiente ciclo evolutivo, el cual continúa con los ajustes o cambios que el usuario realizó y/o con el proceso evolutivo normal. La actividad de ingeniería en cada ciclo de la espiral se puede efectuar con el enfoque de vida clásico o con la creación de prototipos.

El modelo en espiral permite el desarrollo de sistemas y de software a gran escala, ya que en cada ciclo se puede determinar el riesgo y resolverlo de la manera más conveniente, actividad que está apoyada con el uso de los prototipos. Es decir, el modelo en espiral reduce los riesgos en cada etapa, antes de que se conviertan en problemas.

2.2.3. El desarrollo rápido de aplicaciones

El RAD es el desarrollo de software basado en el modelo de la espiral con una estrategia de división del proyecto, principalmente si es grande, en cuadros de tiempo [Rub97]. Un cuadro de tiempo define el plazo en el que se acuerda entregar al usuario final una parte del

proyecto. La totalidad del proyecto se desarrolla y entrega en n fases o cuadros de tiempo, como se ilustra en la siguiente figura.

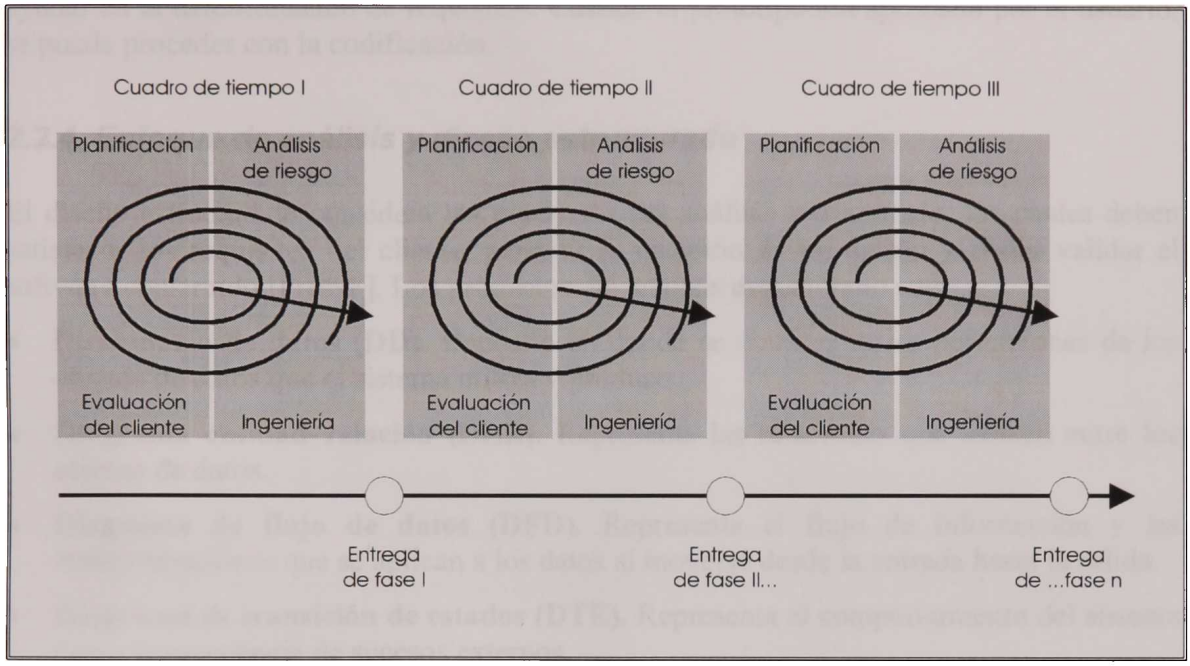


Figura 5. Tres iteraciones de espirales dentro de cuadros de tiempo.

Ventajas y desventajas

En la práctica, el RAD sufre de malas aplicaciones y de abusos porque muchos gerentes y programadores ven el modelo en espiral como un proceso de tres iteraciones de ajuste de "codificación". Esto se traduce a una liberación de "prototipo funcional" y no de un "sistema funcional".

Las ventajas del RAD son:

- Codificación temprana en forma correcta.
- Involucra intensamente a los usuarios.
- Creación temprana de prototipos.
- Implementación en fases.

Las desventajas del RAD son:

- Tendencia hacia la mala codificación temprana.
- Impide la creación de componentes reutilizables a largo plazo.
- Acelera el proceso de documentación.

Hay que hacer notar que la codificación temprana es tanto una ventaja como desventaja. En sistemas donde la complejidad del negocio es baja o nula, se puede codificar sin riesgo, ya que los requisitos se pueden traducir directamente en una codificación. Para aquellos sistemas donde la complejidad es alta, es sumamente recomendable hacer un prototipo para ayudar en la determinación de requisitos. Cuando el prototipo sea aprobado por el usuario, se puede proceder con la codificación.

2.2.4. Enfoque de análisis y diseño estructurado

El diseño estructurado considera los productos del análisis estructurado, los cuales deben satisfacer los requisitos del cliente, permitir la creación de un diseño y poder validar el software construido [Pres98]. Los productos del análisis estructurado son:

- **Diccionario de datos (DD).** Depósito en donde se almacenan las definiciones de los objetos de datos que el sistema utiliza y produce.
- **Diagrama entidad–relación (DER).** Representa las relaciones que existen entre los objetos de datos.
- **Diagrama de flujo de datos (DFD).** Representa el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida.
- **Diagrama de transición de estados (DTE).** Representa el comportamiento del sistema como consecuencia de sucesos externos.

Cada uno de los productos del análisis permite obtener información para desarrollar un modelo de diseño, como se muestra en la Figura 6. La etapa de diseño define:

- **Diseño de datos.** Crea las estructuras de datos necesarias para la implementación del software. El diccionario de datos y el diagrama entidad–relación permiten efectuarlo.
- **Diseño arquitectónico.** Define la relación entre los elementos estructurales del programa. El diagrama de flujo de datos permite establecer dicha relación.
- **Diseño de interfaz.** Permite definir la comunicación entre el propio sistema y con el usuario a partir del diagrama de flujo de datos.
- **Diseño procedimental.** Realiza la descripción de procedimientos requeridos para la operación del sistema. Se puede realizar a partir del diagrama de transición de estados, y de las especificaciones de control y procesos.

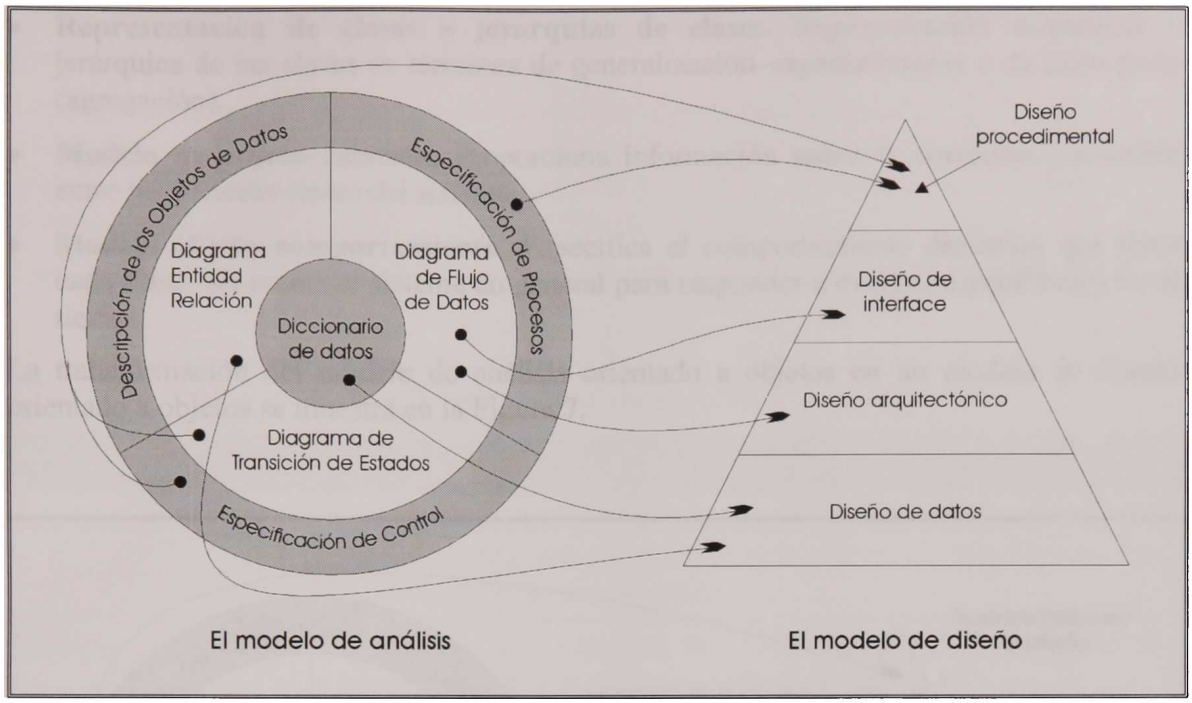


Figura 6. Transformación del modelo de análisis en un modelo de diseño.

En los sistemas cliente/servidor, existe un conjunto de tareas que debe ejecutarse sin interrupción con respecto a la parte del cliente. Dicho conjunto de tareas se define como un proceso elemental de negocios. La descomposición funcional efectuada con los DFD se detiene en el nivel de un proceso de negocios elemental, en lugar de proseguir hasta el nivel de procesos atómicos.

2.2.5. Enfoque de análisis y diseño orientado a objetos

La orientación a objetos se basa en los conceptos de objetos, atributos, partes, clases y miembros. El Análisis Orientado a Objetos (AOO) proporciona los elementos para definir clases y objetos, los cuales reflejan directamente el dominio del problema y las responsabilidades del sistema [Coa91]. El Diseño Orientado a Objetos (DOO) identifica y define clases y objetos adicionales, que reflejan una implementación de los requerimientos [Coa91a].

El proceso de análisis orientado a objetos puede iniciarse con la elaboración de casos de uso, los cuales permiten describir la manera en cómo debe emplearse el sistema orientado a objetos. Posteriormente se puede utilizar la modelación de Clase–Resposabilidad–Colaboración (CRC) y obtener clases, sus atributos y operaciones; así como, la colaboración inicial entre objetos.

A partir de estos métodos, es posible obtener los productos del análisis orientados a objetos, los cuales son los siguientes:

- **Representación de clases o jerarquías de clases.** Representación estructural y jerárquica de las clases en términos de generalización–especialización o de todo–parte (agregación).
- **Modelo de objeto–relación.** Proporciona información sobre la conexión que existe entre las diversas clases del sistema.
- **Modelo objeto–comportamiento.** Especifica el comportamiento dinámico que tiene cada clase; así como, el sistema en general para responder a eventos específicos y en el tiempo.

La transformación del modelo de análisis orientado a objetos en un modelo de diseño orientado a objetos se muestra en la Figura 7.

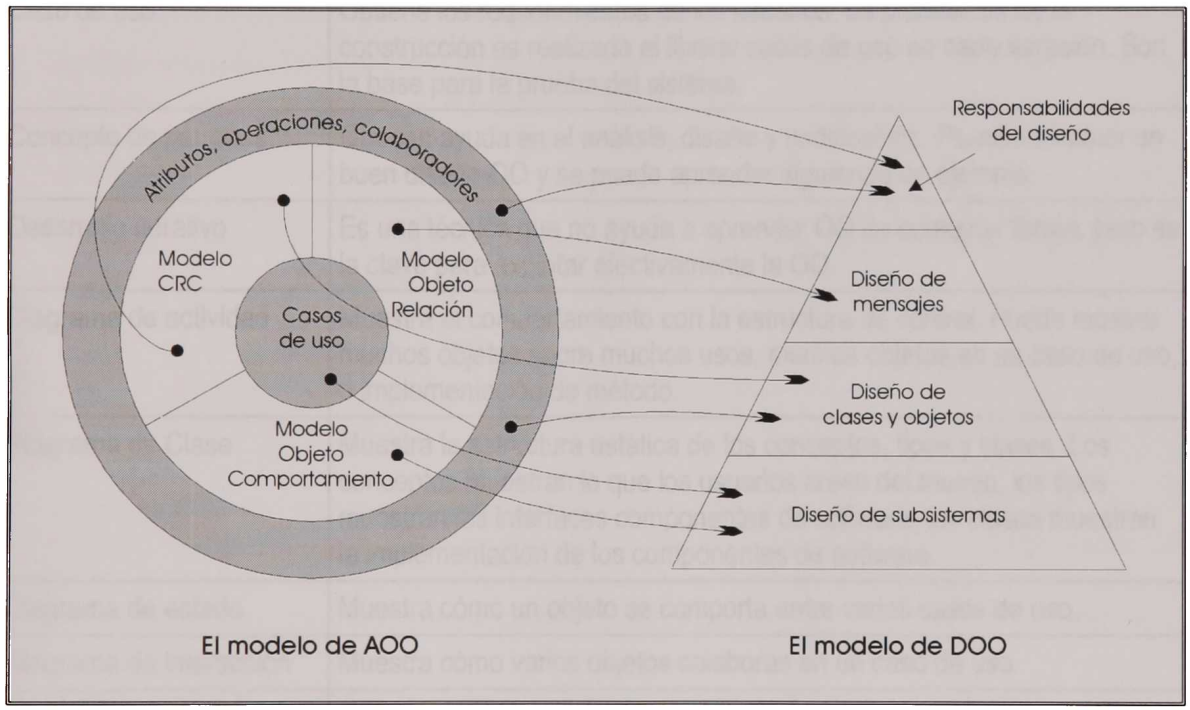


Figura 7. Transformación del modelo de AOO en un modelo de DOO.

El diseño de datos se efectúa en el momento que se representan atributos, el diseño de interfaces al desarrollar un modelo de intercambio de mensajes, el diseño de procedimientos cuando se realiza el diseño de operaciones, y el diseño de arquitectura tiene más que ver con las colaboraciones que con el flujo de control.

2.2.6. El Lenguaje de Modelado Unificado

El Lenguaje de Modelado Unificado (UML: *Unified Modeling Language*) es un lenguaje de modelado estándar para software (un lenguaje para visualizar, especificar, construir y

documentar software) [Jac99]. UML unifica los métodos de Grady Booch, James Rumbaught e Ivar Jacobson.

UML es un lenguaje de modelado, no un método. Los métodos consisten de un lenguaje de modelado y un proceso. El lenguaje de modelado es principalmente una notación gráfica que los métodos usan para expresar el diseño. El proceso es su seguimiento o los pasos que se consideran al hacer un diseño[Fow97]. UML es independiente del proceso.

Las técnicas que UML emplea y el propósito de cada una de ellas se resumen en la Tabla 1.

Tabla 1. Técnicas de UML.

Técnica UML	*Propósito*
Caso de uso	Obtiene los requerimientos de los usuarios. La planeación de la construcción es realizada al liberar casos de uso en cada iteración. Son la base para la prueba del sistema.
Concepto de patrones	Ofrecen ayuda en el análisis, diseño y codificación. Permite efectuar un buen diseño OO y se puede aprender siguiendo un ejemplo.
Desarrollo iterativo	Es una técnica que no ayuda a aprender OO de cualquier forma, pero es la clave para explotar efectivamente la OO.
Diagrama de actividad	Muestra el comportamiento con la estructura de control. Puede mostrar muchos objetos sobre muchos usos, muchos objetos en un caso de uso, o implementación de método.
Diagrama de Clase	Muestra la estructura estática de los conceptos, tipos y clases. Los conceptos muestran lo que los usuarios creen del mundo, los tipos muestran las interfaces componentes de software, las clases muestran la implementación de los componentes de software.
Diagrama de estado	Muestra cómo un objeto se comporta entre varios casos de uso.
Diagrama de interacción	Muestra cómo varios objetos colaboran en un caso de uso.
Diagrama de paquete	Muestra los grupos de clases y las dependencias entre ellos.
Diagrama de producción	Muestra la capa física de los componentes sobre nodos de hardware.
Diseño por contrato	Provee definiciones rigurosas del propósito de operaciones y del estado legal de las clases.
Refactoring	Ayuda a realizar cambios al programa para improvisar estructura.
Tarjetas CRC	Ayuda a obtener la esencia del propósito de las clases. Se diseñaron para que la gente aprendiera a trabajar con objetos. El énfasis en las responsabilidades y su notación las hacen valuables.

2.2.7. Técnicas para la distribución de datos

En los sistemas C/S, los datos pueden existir tanto en el cliente como en el servidor. Al realizar solicitudes de datos es preciso saber la ubicación de los mismos. Para esto, las técnicas empleadas son las siguientes:

- **Extracción manual.** El usuario está facultado para hacer copias de los datos y guardarlos en forma local. Es recomendable su utilización para datos estáticos.
- **De forma instantánea.** Es una alternativa a la extracción manual, ya que de forma automática y periódica se efectúa una copia de los datos que el cliente utiliza. En este caso los datos son relativamente estáticos.
- **Duplicación.** El sistema mantiene varias copias idénticas de los datos. Cada copia se almacena en una localidad (cliente y/o servidor) diferente, lo que resulta en la duplicación de la información [Kor87]. La sincronización y consistencia son esenciales.
- **Fragmentación.** Es el proceso de optimizar los datos distribuidos, de tal forma que sólo los datos que se necesiten se ubiquen de forma local. La forma de hacerlo es: (1) por fragmentación vertical, donde solamente determinadas tablas o columnas están distribuidas físicamente en sitios remotos; (2) por fragmentación horizontal, donde solamente determinados renglones de una tabla están distribuidos físicamente en sitios remotos [Rub97]. La mejor optimización es una combinación de ambas fragmentaciones: la fragmentación mezclada.

2.3. Comentarios

La estructura de los sistemas cliente/servidor hace necesario que se tengan dos niveles importantes de distribución. En primer lugar se tiene la distribución de componentes de software, los cuales definen clientes principales y servidores principales; y en segundo lugar, la distribución de datos que se efectúa principalmente por técnicas de duplicación y fragmentación.

La arquitectura de CORBA permite la comunicación entre objetos clientes y objetos servidores, los cuales deben estar definidos en IDL para garantizar la independencia del lenguaje y de la plataforma.

Los modelos evolutivos e iterativos de ingeniería de software, específicamente el modelo en espiral y la construcción de prototipos, los enfoques tradicionales y orientados a objetos, y los lenguajes de modelado robustos como es el caso de UML, son totalmente aplicables para el desarrollo de sistemas cliente/servidor.

Capítulo 3. Tecnologías y Herramientas para la Construcción de los SII

Los SII se componen fundamentalmente de tres tipos de aplicaciones web:

1. **Aplicaciones de hipertexto estático.** Están caracterizadas por enlaces y páginas estáticas.
2. **Aplicaciones con bases de datos.** Caracterizadas por la creación dinámica de páginas pero con estructura de enlaces estáticos.
3. **Aplicaciones dinámicas.** Caracterizadas por páginas y enlaces dinámicos.

Un SII comprende al menos aplicaciones del tipo centradas en bases de datos, además, cada tipo de aplicación tiene una tecnología implícita, y la tecnología común a las tres es el HTML. Sin embargo, el HTML tiene que ser combinado con una serie de tecnologías y estándares para garantizar la creación y la evolución de los sistemas.

El objetivo del capítulo es mencionar brevemente las principales tecnologías derivadas del SGML, y revisar un par de herramientas que se utilizaron para desarrollar SII reales, a saber: WebSpeed y WebObjects.

3.1. Tecnologías

La mayoría de los documentos sobre la WWW están escritos en un lenguaje de marcación simple llamado HTML. Los documentos HTML contienen etiquetas que especifican cómo se deberán presentar los datos en el *browser*. HTML es realmente un subconjunto de SGML, lenguaje mucho más amplio que fue definido para ayudar a codificar documentos que se visualicen de la misma forma.

Las siguientes secciones describen las principales tecnologías derivadas del SGML que están disponibles para la construcción de SII. Algunas de ellas están en la etapa final de especificaciones.

3.1.1. XML

XML (*eXtensible Markup Language*) se deriva del SGML, y técnicamente es un subconjunto pequeño de SGML con una sintaxis simple. Al igual que HTML (que también

es una instancia de SGML), usa elementos y atributos, los cuales se indican en un documento con el empleo de etiquetas [hLau].

XML, provee el fundamento para la creación de documentos y sistemas de documentos al operar en dos niveles principales:

- Provee una sintaxis para el marcado del documento.
- Provee una sintaxis para declarar la estructura de los documentos.

El siguiente ejemplo muestra la sintaxis de marcado:

```
<FIGURE DESCRIPTION="Canito">
<IMAGE/>
<CAPTION>Mi mascota preferida</CAPTION>
</FIGURE>
```

En la sintaxis de marcado se define una figura con su descripción, la cual tiene una imagen vacía y un título. El siguiente código muestra la estructura del documento, según el ejemplo anterior:

```
<!ELEMENT FIGURE(IMAGE, CAPTION)>
<!ATTLIST FIGURE DESCRIPTION CDATA #IMPLIED>
<!ELEMENT IMAGE EMPTY>
<!ELEMENT CAPTION (#PCDATA)>
```

La estructura del documento indica que el elemento de figura (FIGURE) debe contener un atributo de imagen (IMAGE) y un atributo de título (CAPTION), además puede tener un atributo de descripción. El elemento de imagen debe ser nulo, ya que en caso de existir tendría una serie de atributos para la información de la imagen. El elemento de título (CAPTION) contiene texto, entidades, instrucciones de procesamiento, y cualquier otro texto de XML válido.

XML permite fijar estándares para definir la forma en que deberá aparecer la información en los documentos, lo cual permite reutilizar el contenido en otras aplicaciones y en otros ambientes. Provee una sintaxis básica que se puede usar para compartir información entre diferentes computadoras, diferentes aplicaciones y diferentes organizaciones. Además, XML es considerado como un mecanismo de transporte para la creación de empresas basadas en la Web (WBEM: *Web–Based Enterprise Management*) [hDMT], y en la implementación del Modelo Común de Información (CIM: *Common Information Model*) [hKey].

Las ventajas de XML se pueden resumir en simplicidad, extensibilidad e interoperabilidad, las cuales se describen en las siguientes secciones.

Simplicidad

XML tiene reglas que ayudan a hacer documentos legibles, permite desarrollar de forma correcta y fue construido sobre un conjunto de estructuras básicas. Estas estructuras básicas pueden emplearse para la representación de conjuntos complejos de información, desde contenidos hasta comandos de programas.

Extensibilidad

XML es extensible en dos sentidos. El primero, permite crear a los desarrolladores sus propias definiciones de tipo de documento DTD (*Document Type Definition*) al crear etiquetas que pueden usarse para múltiples aplicaciones, se tiene una extensión propia de XML, debido a que DTD describe la gramática usada por un documento de XML y es opcional. El segundo sentido de extensibilidad consiste en agregar: estilos, enlaces y modos referenciales por medio de una serie de estándares que extienden a XML.

De ésta forma, XML puede utilizar estándares aplicados a HTML, CSS (Cascade Style Sheets) y HTTP, y otros más que están bajo discusión [hLau].

Interoperabilidad

XML puede usarse en una amplia variedad de plataformas y herramientas, soporta un gran número de estándares para codificación, y los analizadores gramaticales están disponibles en Java, C++, JavaScript, Tcl, Python, entre otros. Trabajos anteriores sobre el soporte con clases de Java para extender las capacidades de HTML, los Displets, se pueden encontrar en [hVit].

Apertura

El estándar es por sí mismo completamente abierto, y está disponible en la Web. Los miembros de W3C y los invitados expertos son los únicos que pueden participar en la creación de XML, sin embargo, desde que el estándar fue terminado, los resultados son totalmente públicos [hLau].

3.1.2. XSL

Los documentos XML generalmente tienen información sobre su estructura y su semántica de los datos, pero no tienen información acerca de su visualización. Para solucionar la visualización de los datos se ha propuesto un estándar llamado *eXtensible Style Language* (XSL). Una hoja de estilo XML es esencialmente un grupo de reglas para transformar un documento XML, por medio de mapeos recursivos para formar una estructura de representación, tal como HTML o DHTML [hKes].

3.1.3. Componentes

La tecnología de la Web, hace posible la creación de objetos distribuidos, los cuales deben ser independientes del lenguaje en el que se programaron. Así, los objetos distribuidos se convierten en componentes de software que resultan ser independientes y capaces de actuar en diferentes redes, sistemas operativos y paletas de herramientas.

Propiedades de los componentes

Los componentes tienen una serie de propiedades, las cuales se describen en la siguiente tabla [Orf96].

Tabla 2. Propiedades de los componentes.

Propiedad	*Descripción*
Es una entidad que se puede comercializar	Un componente es una pieza binaria (caja negra) de software autónomo que se puede empaquetar de forma comercial.
No es una aplicación completa	Un componente puede combinarse con otros componentes para formar una aplicación completa. Está diseñado para desempeñar un conjunto limitado de tareas dentro del dominio de una aplicación. Los componentes pueden ser objetos de grano fino, equivalente al tamaño de los objetos en C++; objetos de grano mediano, como los controles de las UI, u objetos de grano grueso, como un applet de Java.
Se le puede emplear en combinaciones múltiples	Como los objetos, un componente puede usarse en múltiples formas. Usualmente los componentes pueden combinarse con otros componentes de la misma familia llamada "serie".
Tiene una interfaz claramente definida	Como los objetos clásicos, un componente puede ser manipulado a través de su interfaz. Éste es el medio por el cual el componente expone su función al mundo exterior. Un componente ofrece en sí mismo un IDL, que se puede emplear para invocar al propio componente o heredar y anular sus funciones. Cabe indicar que la interfaz del componente, similar a la de los objetos es independiente de su implementación. Un componente puede implementarse con el uso de objetos, código de procedimientos o encapsulado del código existente.
Es un objeto interoperable	Un componente puede ser invocado como objeto entre espacios de direccionamiento, redes, lenguajes, sistemas operativos y herramientas. Es una entidad de software independiente del sistema.
Es un objeto extendido	Los componentes son objetos auténticos en el sentido de que soportan encapsulado, herencia y polimorfismo. Sin embargo también deben contar con todas las características asociadas con un objeto comercial autónomo.

3.2. Herramientas

Las herramientas que se usan para la creación de SII tienen características de generación dinámica de páginas con interacción de scripts (guiones o programas) de lenguajes de cuarta generación (4GL) [hBow], orientados a objetos (como Objective–C) y basados en objetos (como JavaScript). Así, los scripts de lenguajes juegan un papel importante en el desarrollo de interfaces, las cuales permiten acceder a las bases de datos en forma simple y portable [hLaz].

Para la generación dinámica de páginas se tienen varias herramientas que ofrecen lenguajes de tercera y cuarta generación con las que se pueden tener acceso a bases de datos robustas.

Estas herramientas las ofrecen compañías como Oracle, Vision 4GL de Unify, Web Element de Neuro Data, Borland, Sybase, Progress, NeXT, entre otros.

Las siguientes secciones describen dos herramientas que se revisaron, WebSpeed y WebObjects, De forma específica, WebSpeed se utilizó para desarrollar algunos proyectos precisos: Sistema de Información de Auditoría, Seguridad Industrial y Protección Ambiental (SIASIPA) y Sistema de Información de la Contraloría Interna (SICI); ambos para PEMEX Refinación. Y el sistema Mex–Money Web (MMW) para Casas de Cambio.

3.2.1. WebSpeed

WebSpeed de Progress Software, es una herramienta de desarrollo y Servidor de Transacciones sobre Internet (ITP: *Internet Transaction Processing*) para la creación y producción de aplicaciones críticas de negocios. Emplea tecnología de cuarta generación (4GL) y soporta SQL en combinación de JavaScript y HTML, así como la compatibilidad con controles Active–X y applets de Java.

WebSpeed, bajo un ambiente de desarrollo basado en *browser*, permite crear aplicaciones Web, o utilizar páginas HTML ya creadas con cualquier herramienta Web de HTML y generar un mapa hacia una base de datos por medio de un lenguaje basado en transacciones, SpeedScript. Las aplicaciones son accesadas desde un servidor Web.

Cada petición desde un cliente Web al ambiente de WebSpeed ejecuta un objeto Web, el cual responde con una página Web que envía al cliente Web original. El código de un objeto Web provee una manera fácil y flexible de generar páginas Web en forma dinámica y pasar información entre páginas Web.

Arquitectura de WebSpeed

La Figura 8 muestra el modelo de aplicaciones de WebSpeed, donde se aprecia el modelo de ejecución y el modelo de desarrollo. En ambos modelos se hace uso de dos componentes compartidos: el Transaction Server y el Workshop, los cuales se definen a continuación.

- **Transaction Server.** Es un motor de producción y desarrollo de aplicaciones basadas en Web que permite ejecutar objetos Web para generar páginas Web.
- **Workshop.** Es una herramienta de desarrollo basada en *browser* para crear aplicaciones Web basadas en transacciones.

Procesamiento de peticiones

El Transaction Server es en realidad un conjunto de procesos que comunican el servidor Web, coordina las peticiones de objetos Web en nombre del servidor Web, ejecuta los objetos Web que atienden las peticiones y accesa el servidor de la base de datos. Los procesos incluyen:

- **El corredor de transacciones.** Es iniciado por el administrador del sistema para coordinar todas las peticiones de los objetos Web realizadas por una aplicación de WebSpeed. Pueden iniciarse varios Corredores.

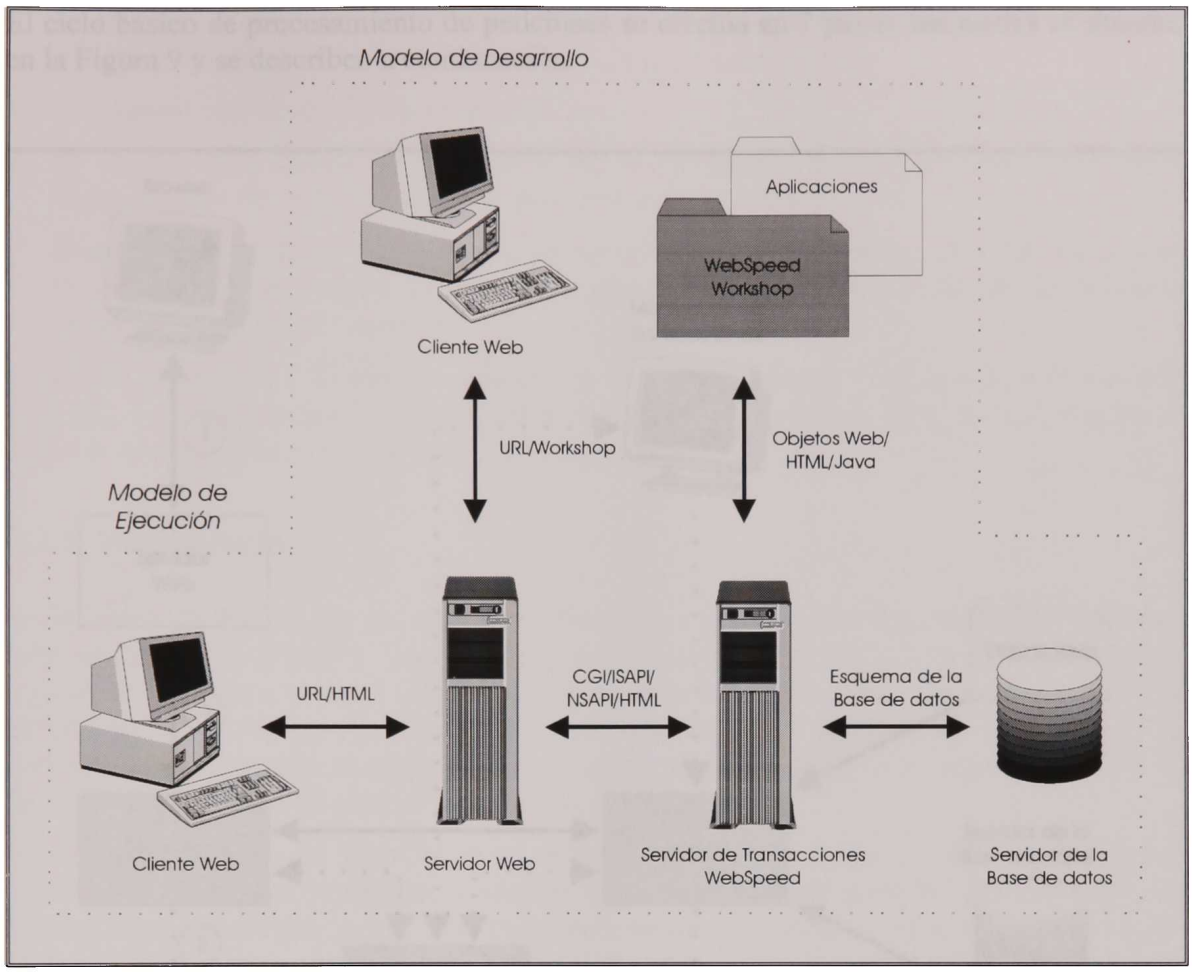


Figura 8. Modelo de aplicaciones WebSpeed.

- **El agente de transacciones.** Producido por el Corredor o iniciado por el administrador del sistema. Cada Agente activo está disponible para ejecutar cualquier objeto Web en la aplicación a menos que se esté ejecutando un objeto Web o que esté asegurado por una transacción de WebSpeed. Al liberarse un Agente, está nuevamente disponible para ejecutar cualquier otro objeto Web.
- **El mensajero o despachador.** Es un programa CGI o un productor de procesos ISAP/NSAPI ejecutados por el servidor Web para atender las peticiones de los objetos Web. El Mensajero termina después de que cada petición es atendida. Los mensajeros ISAPI/NSAPI se llaman y activan entre peticiones y solo ellos soportan la funcionalidad de Despachador.
- **El manejador de transacciones.** Es usado por el administrador del sistema para manejar los Corredores y los Agentes: terminar un Corredor, iniciar y detener un número específico de Agentes, detener un Agente particular, u obtener información de los Agentes activos.

El ciclo básico de procesamiento de peticiones se efectúa en 7 pasos, los cuales se ilustran en la Figura 9 y se describen a continuación.

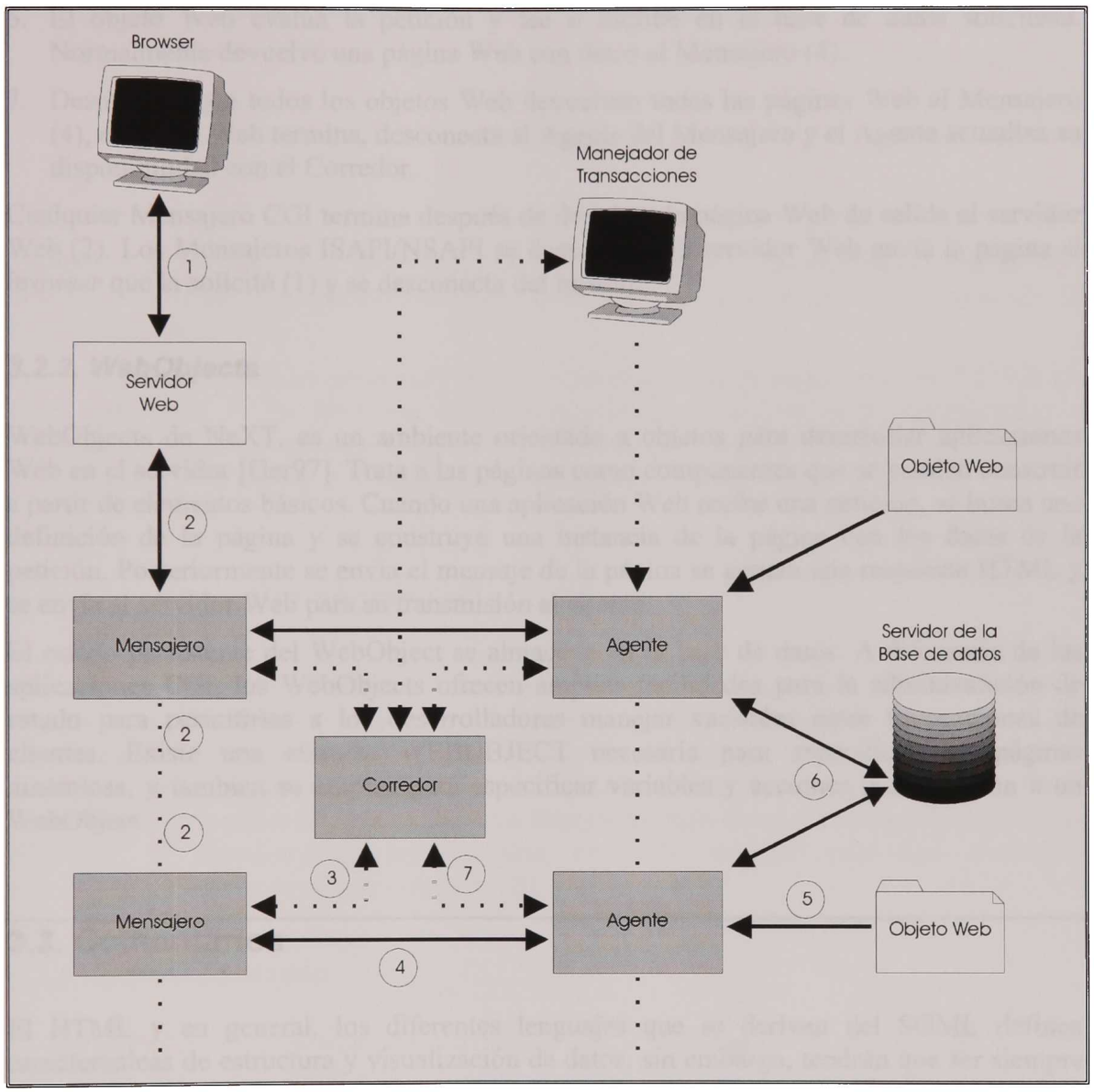


Figura 9. Servidor de Transacciones de WebSpeed.

1. El servidor Web recibe una petición del *browser*.
2. El servidor Web produce un Mensajero CGI o levanta inicialmente un Mensajero ISAPI/NSAPI.
3. El mensajero obtiene el Corredor para un Agente disponible. Si la petición es parte de una transacción persistente, el Corredor provee un puerto del Agente que maneja la transacción.

4. El Mensajero inicia una conexión al Agente devuelto por el Corredor, pasa el nombre del objeto Web y el ambiente del CGI para la petición al Agente.
5. EL Agente ejecuta el objeto Web solicitado.
6. El objeto Web evalúa la petición y lee o escribe en la base de datos solicitada. Normalmente devuelve una página Web con datos al Mensajero (4).
7. Después de que todos los objetos Web devuelven todas las páginas Web al Mensajero (4), el objeto Web termina, desconecta al Agente del Mensajero y el Agente actualiza su disponibilidad con el Corredor.

Cualquier Mensajero CGI termina después de devolver la página Web de salida al servidor Web (2). Los Mensajeros ISAPI/NSAPI se desactivan. El servidor Web envía la página al *browser* que la solicitó (1) y se desconecta del mismo.

3.2.2. WebObjects

WebObjects de NeXT, es un ambiente orientado a objetos para desarrollar aplicaciones Web en el servidor [Ger97]. Trata a las páginas como componentes que se pueden construir a partir de elementos básicos. Cuando una aplicación Web recibe una petición, se busca una definición de la página y se construye una instancia de la página con los datos de la petición. Posteriormente se envía el mensaje de la página se genera una respuesta HTML y se envía al servidor Web para su transmisión al cliente.

El estado persistente del WebObject se almacena en la base de datos. A diferencia de las aplicaciones CGI, los WebObjects ofrecen amplias facilidades para la administración de estado para permitirles a los desarrolladores manejar variables entre invocaciones de clientes. Existe una etiqueta WEBOBJECT necesaria para especificar las páginas dinámicas, y también se emplea para especificar variables y acciones que se pasan a un WebObject.

3.3. Comentarios

El HTML y en general, los diferentes lenguajes que se derivan del SGML definen características de estructura y visualización de datos; sin embargo, tendrán que ser siempre combinados con lenguajes robustos, como SQL y los de tercera y cuarta generación para garantizar la construcción de SII, ya que a través de ellos es posible realizar acceso a bases de datos.

Ésta combinación no es única, ya que incluso es posible realizar una serie de combinaciones entre diversos lenguajes de programación, como lo sería el caso entre HTML y JavaScript, con SQL y/o un lenguaje de programación adicional.

Se discuten principalmente las características de WebSpeed ya que es la herramienta que se utilizó para el desarrollo de algunos sistemas, y además, porque tiene la capacidad de trabajar e interactuar con la combinación antes mencionada y otras más: HTML, JavaScript, ActiveX, Java, SQL y 4GL.

Capítulo 4. Metodología de Diseño Propuesta

El análisis de los SII puede establecerse a partir de enfoques tradicionales u orientados a objetos, por lo cual el diseño debe mantener el enfoque del análisis más la consideración de la naturaleza de la Web.

El objetivo del capítulo es presentar y explicar una propuesta de diseño para los SII, basada en la naturaleza de la Web con un soporte robusto de los enfoques tradicionales y orientados a objetos, y considerando las aportaciones que ofrecen trabajos previos sobre el tema (citados en la página 1) y algunas restricciones que deben tomarse en cuenta.

4.1. El proceso de desarrollo.

El proceso de desarrollo para los SII, se basa en el modelo de la espiral y en la utilización de prototipos. El proceso evolutivo se efectúa desde dos perspectivas, un macroproceso que representa la secuencia evolutiva del modelo de la espiral, y un microproceso que representa cada uno de las iteraciones de la espiral y que de forma específica comprende pasos del ciclo tradicional, los cuales son integrados bajo la construcción de prototipos. Entonces, el macroproceso implica una serie de microprocesos que describen adecuadamente el desarrollo evolutivo de SII.

4.1.1. El macroproceso

El macroproceso incluye aquellas actividades que necesitan cubrirse en su totalidad para desarrollar un sistema, como son:

- Establecer el plan general del proyecto.
- Recopilación de requisitos.
- Declaración de problemas y oportunidades.
- Definición de objetivos, criterios de evaluación y opciones de solución.

De esta forma, el macroproceso define las siguientes actividades:

- **Conceptualización.** Contempla los enfoques tradicionales y/o enfoques orientados a objetos los cuales se deben retroalimentar para obtener la mejor claridad de las necesidades de los usuarios.
- **Composición.** Se efectúan las transformaciones necesarias de los enfoques empleados para obtener las representaciones adecuadas en las cuales se basará la calidad del sistema. Las representaciones distinguen interfaces, datos, procesos y componentes.
- **Evolución.** Codificación de las representaciones establecidas, según las especificaciones de los diseños.
- **Gestión.** Cambios específicos del sistema, y eliminación de errores persistentes.

4.1.2. El microproceso

En el microproceso se llevan a cabo actividades de análisis, diseño y construcción del sistema, aunque son intencionadamente borrosas, razón por la cual se tiene una estrecha relación con las actividades de los prototipos. Los microprocesos corresponden normalmente a módulos o submódulos del sistema, los cuales se desarrollan por etapas. El microproceso engloba las siguientes actividades:

- **Análisis de requisitos.** Validación de requerimientos detectados desde el macroproceso con la incorporación de prototipos de interfaz.
- **Diseño y construcción.** Considera el diseño y construcción de interfaces (funcionalidad y navegación), base de datos (distribución), procesos y componentes (ubicación en cliente y/o servidor).
- **Evaluación y refinamiento.** Interacción con el usuario por medio de la prueba de uso y ajustes al nivel de módulo o submódulo, según sea el caso.

4.2. Actividades de análisis para los SII

Los principios básicos de análisis, los métodos de modelado tradicional y orientados a objetos son aplicables a los sistemas cliente/servidor y por consecuencia a los SII. Los componentes de interacción/presentación con el usuario incrementan la importancia de diseño de interfaces de usuario, razón por la cual desde la etapa de análisis es necesario enfatizar sobre las interfaces primarias involucradas en el sistema.

Entonces, las actividades de análisis para SII son las mismas que se efectúan de acuerdo al enfoque seleccionado con una especial atención hacia la obtención de interfaces primarias, es decir, las interfaces de usuario que inicialmente se detectan para el sistema. En esta sección solamente se hablará sobre el cómo obtener las interfaces primarias, lo cual se describe como sigue:

- **Interfaces obtenidas del enfoque tradicional.** El diagrama de flujo de datos debe tener anotaciones de mecanismos de transporte. Los mecanismos de transporte permitirán obtener las interfaces correspondientes, no obstante, hay que considerar que

los diagramas E–R y de transición de estados aportan elementos necesarios para detallar interfaces.

- **Interfaces obtenidas del enfoque orientado a objetos.** Los casos de uso y los elementos que los integran (descripciones, requisitos, glosario) permiten obtener las interfaces que son necesarias.

Para apoyar la obtención de las interfaces primarias, es recomendable hacer uso de los prototipos conceptuales (de papel) y de las pruebas de uso. Esto permitirá también complementar las especificaciones necesarias de las interfaces en general, así como las rutas de navegación correspondientes.

Una especificación de interfaz (bajo un contexto web) debe contener los siguientes elementos:

- **Entidad web.** Nombre de la interfaz a la cual se hace referencia.
- **Acción.** Nombre de la acción que se realiza dentro de una entidad web.
- **Vínculo.** Es la entidad web involucrada dentro de la acción.

Las acciones siempre corresponden a llamados de interfaces. Una interfaz puede tener varias acciones, en tal caso es necesario anotar los vínculos correspondientes a cada una de las acciones. Pueden existir acciones que hagan referencia a la misma interfaz. La siguiente tabla muestra un ejemplo genérico de la especificación de interfaz.

Tabla 3. Ejemplo genérico de una especificación de interfaz bajo contexto web.

Entidad web	*Acción*	*Vínculo*
EWR	Obtener información general.	EW1
EWR	Registrar los datos del solicitante.	EW2
EW1	Regresar.	EWR
EW2	Registrar datos.	EW3
EW3	Aceptar e ingresar.	EW4
EW4	Realizar operación 1	EW5
EW4	Realizar operación 2	EW6
EW6	Ordenar por concepto.	EW6

4.3. El diseño para los SII

El diseño de los SII comienza con la transformación de las especificaciones de interfaz en una arquitectura bajo la cual el sistema estará soportado. Una vez establecida la

arquitectura, se procede con el diseño de la base de datos relacional y posteriormente se desarrolla una serie de actividades para obtener y representar la funcionalidad de las interfaces involucradas, establecer los procesos de interfaz y de negocios. Los pasos genéricos del diseño para SII se resumen en la siguiente tabla.

Tabla 4. El diseño global de Sistemas de Información Intranet.

Paso	*Descripción*
1. Arquitectura	Diseñar la arquitectura del sistema a partir de las especificaciones genéricas de interfaz.
2. Datos	Diseñar la base de datos relacional.
3. Interfaz	Diseñar la interfaz, sobre la base de elementos estáticos y dinámicos.
4. Procesos	Diseñar los procesos intranet.

4.4. Diseño de la arquitectura del sistema

Al igual que los sistemas C/S tradicionales, la arquitectura tecnológica de los SII se define a partir de los requerimientos y de los modelos de análisis. Los distintos tipos de componentes que definen las capas de software, la distribución de los mismos, el middleware y los corredores de solicitudes de objetos permiten obtener una distribución adecuada de datos y procesamiento para la arquitectura C/S.

La recopilación de estadísticas a partir de los modelos de información y de eventos, proporciona información para realizar la estimación del tamaño de la base de datos y de las capacidades de respuesta necesarias.

El objetivo principal del diseño de un SII es la creación del a arquitectura del sistema. Bajo un ambiente web, la arquitectura del sistema es un esqueleto que representa a las entidades web (páginas) y los vínculos correspondientes. A partir de esto se realizan actividades más detalladas de diseño.

4.4.1. Diagrama de hipervínculos

La especificación de interfaz elaborada en la fase de análisis, es la base para el inicio de la construcción de la arquitectura del sistema. Dicho proceso inicia con la transformación de cada uno de los renglones que contiene la tabla de especificación, en dos rectángulos y una línea direccionada que los conecta. Los rectángulos corresponden a la entidad web y al vínculo, los cuales quedan unidos o asociados por medio de la línea direccionada que parte de la entidad web hacia el vínculo y que representa a la acción.

La estructura obtenida contiene todas las entidades web (EW), acciones y vínculos correspondientes. El diagrama resultante del proceso de transcripción se llama Diagrama de Hipervínculos (o simplemente Diagrama H), el cual representa la arquitectura web del sistema.

Notación del diagrama H

Los elementos notacionales del diagrama H se listan a continuación, y la Figura 10 los muestra en forma gráfica.

- Un rectángulo etiquetado, que representa una entidad web (EW) y su nombre respectivo.
- Un rectángulo etiquetado con borde grueso, que representa una EW que contiene o es contenedora de otras EW.
- Una línea con punta (flecha) que une o comunica a una EW Padre con su EW Hijo, es decir, una EW que contiene una referencia o localización a otra EW.
- Una línea con punta en ambos extremos que representa una referencia o locación entre dos EW que se llaman mutuamente.
- Una arroba (@) que representa una referencia o localización a ella misma. Va antes del nombre de la EW.

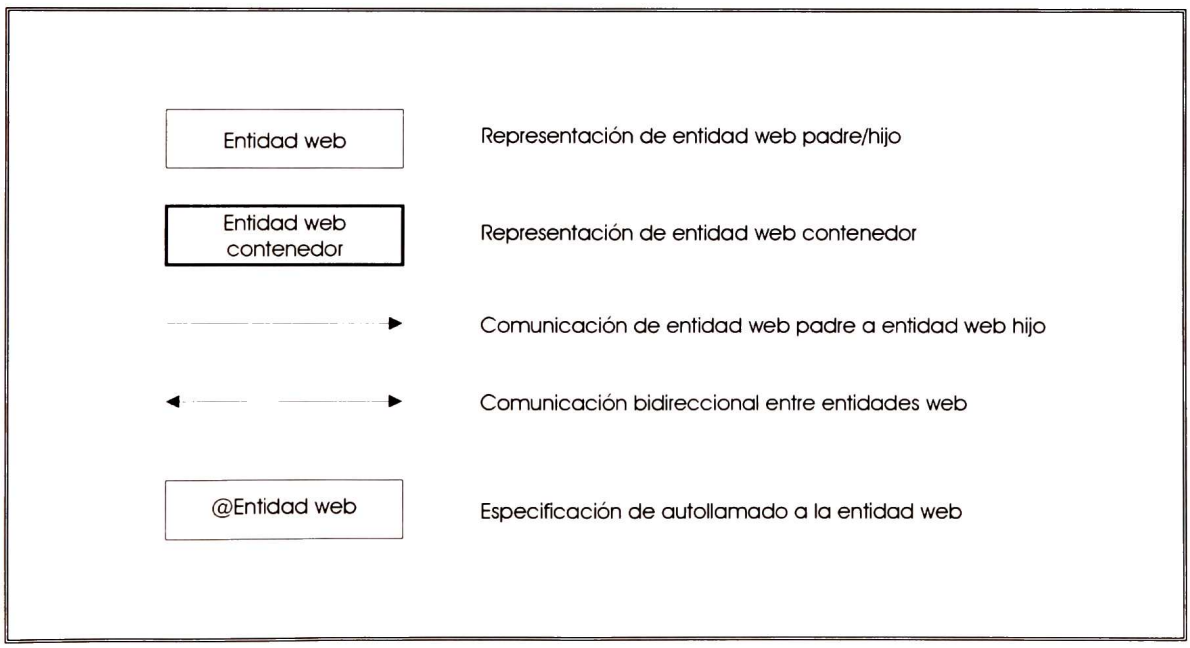


Figura 10. Notación básica para los diagramas H.

Características del diagrama H

Las características que el diagrama H presenta son las siguientes:

- **Representa la organización estructurada o modular del sistema.** Las organizaciones de las páginas pueden ser jerárquicas, lineales, lineales con alternativas o combinaciones entre ellas que pueden verse como una maraña o telaraña. El diagrama H refleja el grado de organización estructurada o modular que el mismo sistema lleva de forma implícita, lo cual permite obtener una jerarquía de EW no muy común en la naturaleza propia de la Web.
- **Representa los hipervínculos de las páginas.** Cada una de las líneas direccionadas que aparece en el diagrama H son las ligas que existen entre las EW. Las líneas doblemente direccionadas indican que entre las dos EW hay ligas de una a otra y viceversa.
- **Representa la navegación del sistema.** El diagrama H es por sí mismo un diagrama de navegación, ya que por medio del seguimiento de las direcciones de los hipervínculos, es posible obtener las rutas de navegación.
- **Representa las secuencias de interfaces.** También, al seguir las direcciones de los hipervínculos, es posible saber cuáles son las interfaces que se llaman, las cuales hacen posible establecer las secuencias de interfaces que satisfacen los objetivos del usuario.

La siguiente figura muestra el diagrama H resultante de la especificación genérica listada en la Tabla 3.

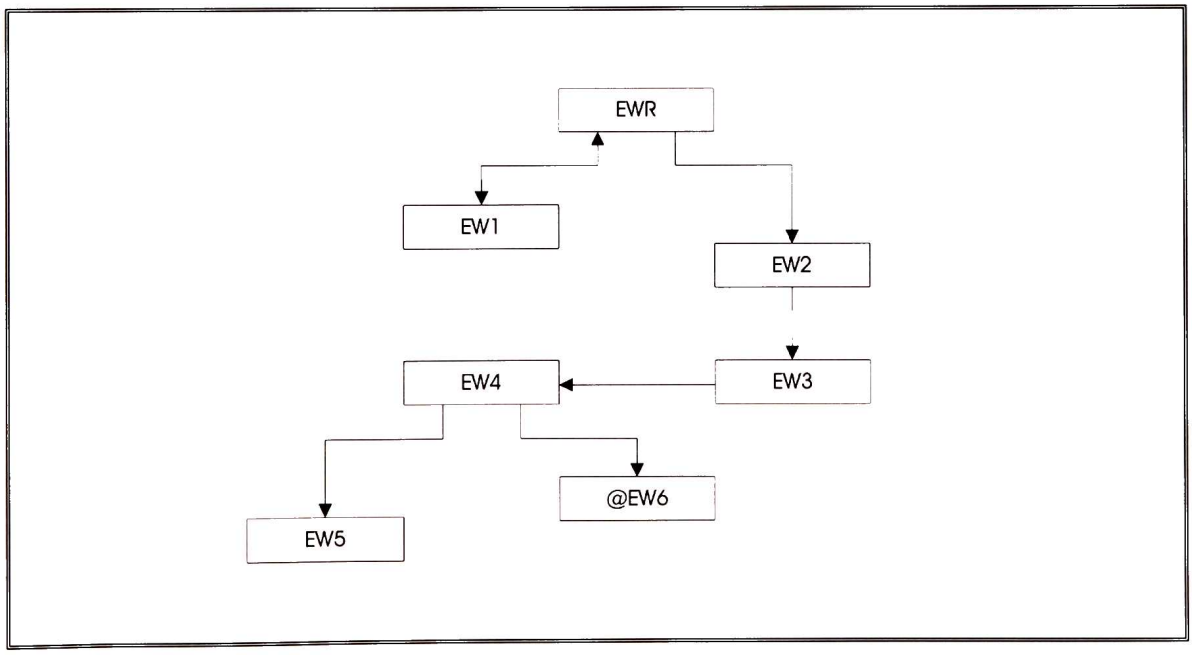


Figura 11. Diagrama H obtenido de la especificación genérica.

Sintaxis del diagrama H

Las reglas existentes para el diagrama H garantizan en lo posible una representación jerárquica de la estructura del sistema. Las reglas son las siguientes:

1. Cuando se tiene un hipervínculo de una EW Padre nueva a una EW Hijo ya creada, se puede repetir el nombre de la EW con la terminación "/n", donde n es un número consecutivo de acuerdo a las EW previas.
2. Cuando una EW Padre tiene hipervínculos a varias EW Hijos, es muy probable que esas EW Hijos puedan ser ubicadas simultáneamente como una sola EW, lo que trae como consecuencia que la EW Padre sea un contenedor (por ejemplo, un frameset de HTML). Para diferenciar este tipo de casos, de otras EW Hijos que se llaman por diferentes acciones, se emplea un cuadro con borde grueso. Es recomendable que las EW Hijos de un contenedor se coloquen a un mismo nivel horizontalmente, y las que no pertenecen a un contenedor, se coloquen de forma escalonada.
3. En caso de que exista una situación como la del punto 2, y con la condición adicional de que alguno de los hijos del contenedor tenga más hiperniveles inferiores, en los cuales llamen a la Padre contenedor, es recomendable también el aplicar la regla 1.
4. Cualquier acomodo horizontal adicional al primero, no es indicativo de otro contenedor, sino EW u operaciones que se llaman según las condiciones dadas en el propio contenedor.

En la Figura 12 se muestra un ejemplo de un diagrama H refinado donde es posible identificar los casos donde se aplican las reglas definidas.

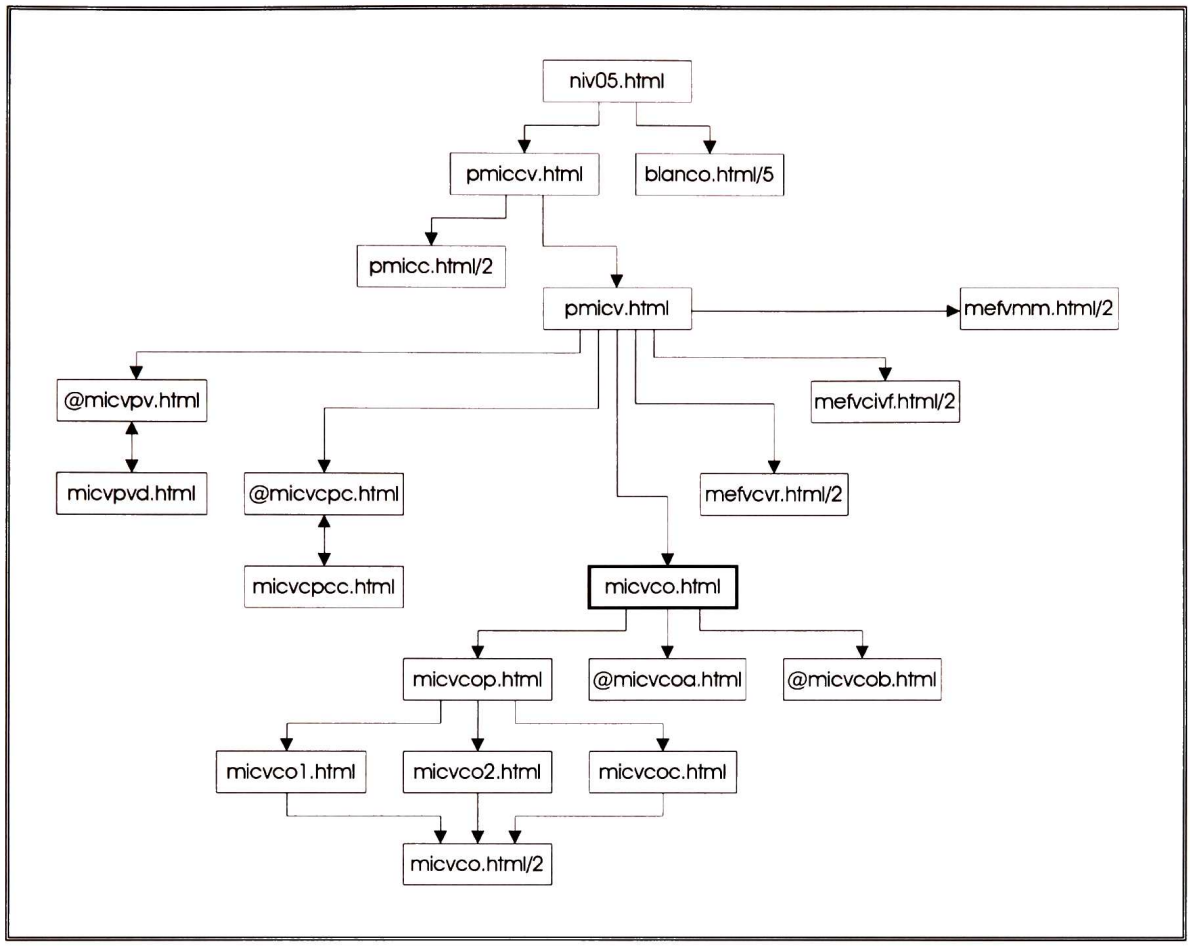


Figura 12. Ejemplo de diagrama H con representación jerárquica.

En la Figura 12 se pueden observar varios ciclos de navegación establecidos desde la EW llamada micvco.html; un ciclo tiene la secuencia siguiente:

micvco.html → micvcop.html → micvco1.html → micvco.html

Otros ciclos más pequeños están establecidos desde la EW de micvpv.html:

micvpv.html → micvpvd.html → micvpv.html

micvpv.html → micvpv.html → micvpvd.html

El acomodo de las EW en el diagrama H permite especificar adecuadamente configuraciones de contenedores, por ejemplo, las tres EW a las que llama la EW micvco.html (micvcop.html, micvcoa.html y micvcob.html). Específicamente, HTML tiene un mecanismo contenedor llamado frameset, y en este caso, para identificarlo posteriormente con facilidad, se engrosa la línea del rectángulo.

Lo anterior demuestra cómo un diagrama H representa la arquitectura web de un sistema, la comunicación entre las EW y la navegación correspondiente entre las EW como en ellas mismas.

4.5. Diseño de la base de datos

El diseño de la base de datos sigue el tradicional método relacional, sin importar el enfoque empleado para el desarrollo del sistema. Los elementos de entrada para el diseño de la base de datos son:

1. El esquema conceptual (diagrama E–R).
2. Una descripción del modelo lógico objetivo y las restricciones.
3. Datos de carga, es decir, la población de la base de datos y el conocimiento de consultas y las transacciones que se realizan en la base de datos, junto con su frecuencia (estadísticas).
4. Criterios de rendimiento como el tiempo de respuesta, el espacio de la base de datos y la utilización de la CPU o el tiempo de E/S.
5. Preferencias del diseñador.

El diseño de la base de datos comprende las siguientes actividades clasificadas como dependientes e independientes del modelo:

- Independientes del modelo.
  - ✓ Toma de decisiones sobre el almacenamiento o no de datos derivados.
  - ✓ Aspectos sobre la eliminación de jerarquías de generalización.
  - ✓ La partición o fusión de entidades y relaciones.
  - ✓ La selección correcta de claves primarias.
- Dependientes del modelo.
  - ✓ Transformaciones de relaciones uno a uno, uno a muchos y muchos a muchos (entidades asociativas).
  - ✓ Transformación de relaciones n–arias y recursivas.

4.5.1. Correspondencia con un enfoque orientado a objetos

Bajo el enfoque orientado a objetos hay que tener en cuenta la correspondencia de clases y asociaciones con las tablas del modelo relacional.

Algunas reglas para la correspondencia de clases y asociaciones (incluso clases de agregación) a tablas son las siguientes [Boo96]:

- Cada clase se hace corresponder con una o más tablas.
- Cada asociación muchos a muchos se hace corresponder con una tabla distinta.
- Cada asociación uno a muchos se hace corresponder con una tabla distinta o puede insertarse como una clave externa.

Para la correspondencia de jerarquías clase/subclase se sugiere una de tres alternativas:

- La superclase y cada subclase se hacen corresponder con una tabla.
- Los atributos de la superclase se repiten en cada tabla (y cada subclase se corresponde con una tabla distinta).
- Elevar todos los atributos de las subclases hacia el nivel de la superclase (y tener una tabla para toda la jerarquía superclase/subclase).

4.6. Diseño de la interfaz de usuario

La interfaz de usuario es el mecanismo a través del cual se establece un diálogo entre el sistema y el usuario. La consideración de los factores humanos es de gran importancia, y para integrarlos de manera adecuada al sistema es necesario emplear un mecanismo que permita reunirlos a la brevedad posible. Tal mecanismo es un prototipo de interfaz.

4.6.1. Prototipo conceptual de interfaz

Las especificaciones de la interfaz realizadas en la etapa de análisis deben cumplir con los objetivos de los usuarios y, a partir de ellas, se elabora un prototipo en un ámbito conceptual. Un prototipo conceptual se realiza con tecnología básica, la cual permite a parte de los bajos costos, la creación y la modificación rápida de las representaciones de las pantallas cuyo contenido se centra en el contenido de información de la ventana y de los eventos del negocio [Rub97].

En caso de necesitarse, el prototipo conceptual puede revisarse con los usuarios. Para lo cual será necesario desarrollar y efectuar pruebas de uso. La revisión debe hacerse de manera personal, para identificar las reacciones de los usuarios, y hacer anotaciones sobre el desempeño y también de sus críticas. La interacción se puede hacer a través de proyecciones o directamente sobre las hojas de papel, con la simulación de las operaciones respectivas.

Técnica de revisión del prototipo conceptual

Las técnicas de revisión llevan a un proceso de desarrollo iterativo basado en el modelo de la espiral. El desarrollo iterativo incluye un medio para la retroalimentación con el usuario. Este proceso define las siguientes etapas [Kas96] mostradas en la tabla siguiente.

Tabla 5. Técnicas de revisión para el modelo en espiral.

Etapa	*Actividad*
1. Planeación.	• Aquí se crea un diseño de la interfaz de tal forma que se pueda presentar a los usuarios finales.

Etapa	*Actividad*
2. Implementación.	• Se presenta el prototipo de la interfaz a los usuarios finales para recibir una retroalimentación para el diseño.
3. Medición.	• Se realiza una evaluación sobre los aspectos que se aprendieron de la fase de implementación. • No es posible hacer grandes mediciones si el proceso no es conducido por la retroalimentación del usuario. • La retroalimentación se obtiene con las descripciones verbales que se le hacen al usuario final. • Lo más importante es medir la efectividad de una implementación antes de proceder con la misma.
4. Aprendizaje.	• Se realiza una refinación de las necesidades de los usuarios finales, las cuales llevan a improvisar una futura implementación.

Las etapas de planeación e implementación definen el desarrollo del prototipo, mientras que las etapas de medición y aprendizaje definen la prueba de uso. El prototipo y la prueba de uso son las dos herramientas con las que se lleva a cabo el proceso de desarrollo iterativo y que dará como resultado un prototipo de interfaz aprobado y validado por el usuario final.

El prototipo de papel involucra un mínimo esfuerzo y permite participar tanto a miembros del equipo como a usuarios finales. La prueba de uso permite medir las respuestas y reacciones de los usuarios finales, y permite llevar control de las fases de medición y aprendizaje. El desarrollo iterativo del prototipo se ilustra en la siguiente figura.

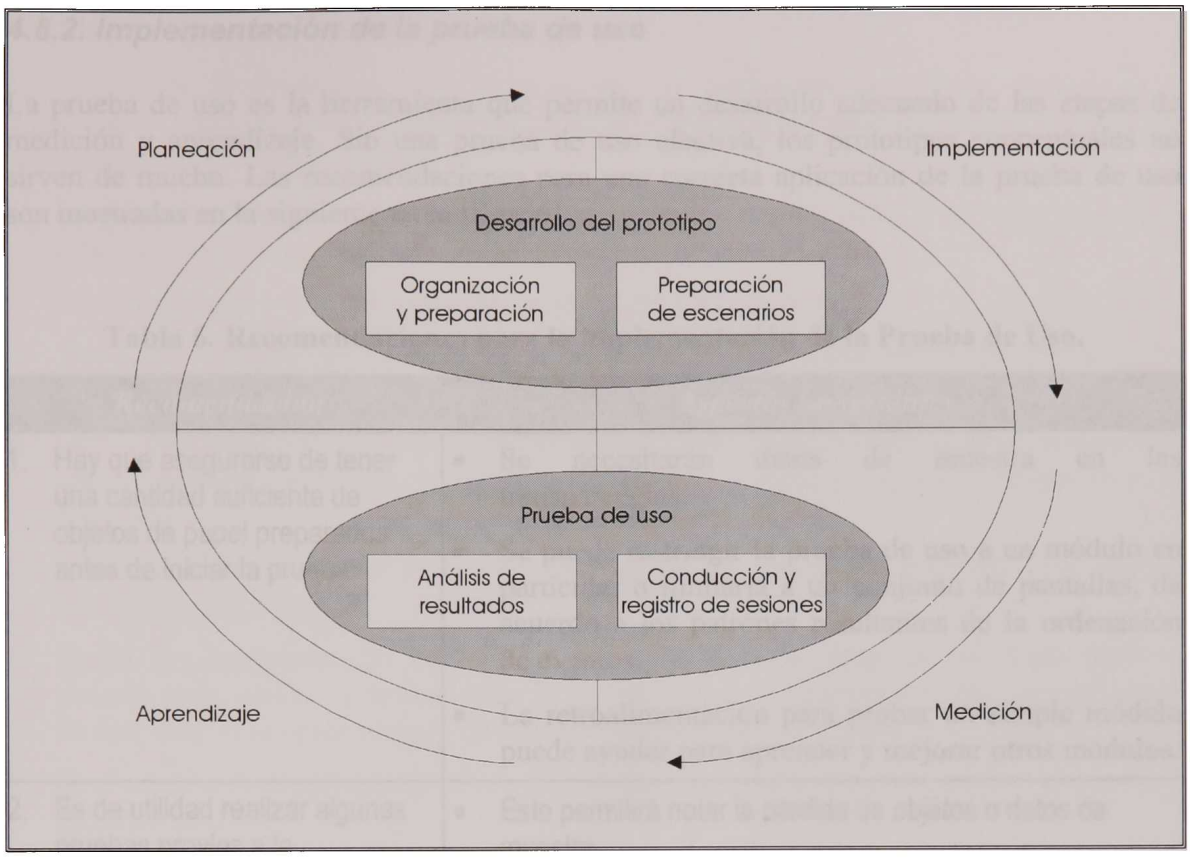


Figura 13. Proceso iterativo para el prototipo conceptual.

Elementos del prototipo conceptual

Los elementos que integran el prototipo conceptual son:

- Modelos de papel para cada objeto o elemento de una interfaz de usuario, por ejemplo listas desplegables, cajas de texto, botones de radio, casillas de verificación, páneles de navegación, etc.
- Modelos de papel para las ventanas, contenedores, marcos y cuadros de diálogo.
- Transparencias de color para simular fondos, botones o elementos de menús.
- Utilizar transparencias para simular vistas con ejemplos de muestra.
- Emplear piezas separadas de papel para arrastrar, colocar y soltar objetos.

Todos los elementos se combinan para obtener una representación de la interfaz, y según se desarrollen las pruebas, el facilitador tiene que cambiar constantemente los modelos para lograr la simulación de las operaciones especificadas por el usuario final. Hay que tener cuidado cuando se represente un dato simple, ya que puede quedar perdido entre todos los demás objetos de la interfaz.

4.6.2. Implementación de la prueba de uso

La prueba de uso es la herramienta que permite un desarrollo adecuado de las etapas de medición y aprendizaje. Sin una prueba de uso efectiva, los prototipos conceptuales no sirven de mucho. Las recomendaciones para una correcta aplicación de la prueba de uso son mostradas en la siguiente tabla [Kas96].

Tabla 6. Recomendaciones para la implementación de la Prueba de Uso.

Recomendación	*Aspectos importantes*
1. Hay que asegurarse de tener una cantidad suficiente de objetos de papel preparados antes de iniciar la prueba.	• Se necesitarán datos de muestra en las transparencias. • Se puede restringir la prueba de uso a un módulo en particular o limitarla a un conjunto de pantallas, de acuerdo a los patrones resultantes de la ordenación de eventos. • La retroalimentación para probar un simple módulo puede ayudar para aprender y mejorar otros módulos.
2. Es de utilidad realizar algunas pruebas previas a la presentación con los usuarios finales, para verificar que la terminación sea correcta.	• Esto permitirá notar la pérdida de objetos o datos de muestra.
3. Realizar una lista de los usuarios finales que participarán.	No hay que forzar el ejercicio a los usuarios finales, sino permitirles la participación en el momento y tiempo oportuno.
4. Preparar a los usuarios finales para el ejercicio. Ellos sabrán que ésta es su oportunidad para participar en el proceso de diseño.	• Explicar las reglas del ejercicio y cómo se beneficiarán con el resultado de la aplicación.
5. Preparar hojas de anotación de las tareas que los usuarios finales intentarán usar en el prototipo.	• Seleccionar tareas relacionadas a cada pantalla o conjunto de pantallas y crear anotaciones que indiquen los pasos necesarios. Esto será también corroborado con las ordenaciones de los eventos. • Todo se preparará con datos de ejemplo que se usarán en dicha tarea, lo que requerirá tener algunas transparencias de avance, para que los usuarios mantengan la atención en el producto, no en los datos.

Reglas para la prueba de uso

Las reglas de la prueba de uso aseguran que se desarrolle una sesión productiva y significativa, tanto para los analistas/diseñadores como para los usuarios finales. Para tal fin es necesario seguir con ciertas reglas:

1. Los usuarios finales usarán su dedo como puntero y clic del ratón, y hay que animarlos a hablar sobre lo que están haciendo con el ejercicio.
2. Asegurarse que todos los materiales están listos sobre la prueba.
3. Tener un miembro del equipo de prototipos asignado a tomar notas del ejercicio. Se pueden usar alternativamente cintas de audio y video para revisarlas posteriormente.
4. Uno o más miembros del equipo de prototipos simularán a la computadora presentando nuevas pantallas, pop–ups, y ejemplos, de acuerdo a como los usuarios finales usen el prototipo.
5. Evitar tener demasiados miembros del equipo de prototipos mirando, esto podría intimidar a los usuarios.
6. Todos los miembros del equipo de prototipos deberán estar en completo silencio durante cada sesión de anotaciones. A menos que la aplicación requiera instrucciones de voz, ésta es la única manera de simular efectivamente el ambiente del usuario.
7. Si el usuario final navega en donde quiera, hay que prepararse para abortar la prueba y empezar otra.

Agenda de conducción

Una agenda de conducción es una serie de pasos que permiten llevar a cabo una prueba de uso. Cuando es la primera sesión, permite comunicarles a los usuarios finales los propósitos y reglas de la prueba y conocer a los miembros del equipo. Los puntos tradicionales de la agenda de conducción son:

1. Bienvenida a los probadores y descripción de los objetivos de la sesión.
2. Presentar a los miembros del equipo.
3. Tener la prueba de introducción y describir el trabajo funcional relacionado al software revisado.
4. Mostrar una sesión de ejemplo que efectúen los miembros del equipo y mostrar también un ejemplo de anotaciones. Indicar también los propósitos de las anotaciones.
5. Explicar las reglas de la prueba de uso.
6. Reenfatizar que la prueba está diseñada para identificar defectos en el prototipo conceptual.

Las características que se obtienen en las anotaciones resultantes de la prueba de uso son muy similares a las siguientes:

- El usuario tuvo dificultad en la pantalla principal, la opción del botón no está suficientemente clara.

- Hay confusiones, se requieren muchos clics, el usuario se pierde.
- El usuario hizo sólo un clic sobre un objeto, cuando debió hacer un doble clic.
- No está claro para el usuario, si el elemento que se agregó se salvó en la base de datos. No hay confirmación.
- En las notas no se preguntó al usuario si es necesario regresar a la pantalla principal. Puede requerirse mejorar la nota para la siguiente sesión.

Las grabaciones y videocintas proporcionan detalles para los aspectos de retroalimentación. Es aconsejable llevar registro de fecha y hora de las sesiones realizadas para revisiones futuras, ya sea para el mismo proyecto o para proyectos futuros.

4.6.3. Especificación conceptual de la interfaz de usuario

La especificación conceptual de la interfaz de usuario (UI) permite:

- Documentar y revisar las ideas de la interfaz sin necesidad de codificar para evaluar.
- Dividir la aplicación entre varios programadores para su construcción.
- Integrar código fácilmente, ya que el trabajo se realiza a partir de una especificación coherente.
- Evaluar la aplicación en forma independiente.
- Una entrega más rápida de un producto terminado de alta calidad.

Sin la especificación, se obtienen cuellos de botella ya que todo el diseño se queda en la cabeza del programador, no se puede evaluar el esfuerzo requerido y cualquier intento para probar el producto terminado está en gran desventaja.

Contenido de la especificación conceptual

La especificación de la UI contiene los elementos listados en la siguiente tabla.

Tabla 7. Elementos de la especificación conceptual de UI.

Elemento	*Descripción*
Entidad web	Nombre de la entidad web que representa el prototipo.
Descripción del prototipo	Descripción sobre las funciones de características reunidas.
Elementos	Listado de todos los elementos que componen el prototipo.
Acciones	Descripción de todas las acciones involucradas en el prototipo.
Vínculos	Entidades web involucradas en las acciones correspondientes.
Campos	Anotaciones sobre los campos que se han identificado.

4.6.4. Prototipo detallado de interfaz

Una vez que el prototipo conceptual se ha cubierto, es posible construir un prototipo detallado con alguna herramienta RAD. Como ya se conocen todos los elementos que integran las UI, el objetivo del prototipo detallado es concentrarse en la funcionalidad de la interfaz, y conocer los aspectos de implementación.

El proceso de revisión con los usuarios es prácticamente igual que en la etapa de diseño conceptual, la diferencia radica en la fase de planeación, donde en vez de crear un diseño, se proporciona funcionalidad a los controles que integran la interfaz. Las pruebas de uso se modifican para que el usuario utilice el prototipo y evalúe la funcionalidad. El proceso iterativo culminará con una interfaz de usuario que satisfaga los objetivos de los usuarios.

Regla de oro para la prueba de la UI

A medida de que se concluyen módulos, es posible efectuar pruebas para garantizar la calidad del sistema. Cada programador debe ser responsable de probar su código individualmente, pero para la evaluación completa del módulo e incluso del sistema completo, existe una sola regla de oro: toda aplicación debe ser probada por alguien diferente al programador original antes de que sea liberada para los usuarios.

4.6.5. Documentación del prototipo detallado

El prototipo detallado tiene que ser también documentado. La documentación incorpora la especificación conceptual refinada, la cual se muestra en la Tabla 8.

Tabla 8. Especificación para el prototipo detallado.

Elemento	*Descripción*
Panorama del sistema	Descripción breve sobre el objetivo y la función del sistema completo.
Panorama de la aplicación	Descripción para cada aplicación del sistema, que define las características disponibles dentro de la aplicación.
Descripción de la ventana	Descripciones sobre la función y características de ésta, en forma tal que el usuario potencial pueda comprender el comportamiento del diseño.
Validación de datos.	Define las reglas para validar los datos que el usuario introduce, con los mensajes adecuados para avisar al usuario.
Validación de errores.	Define la forma en que los errores se tratan para presentar avisos acordes al usuario.
Reglas del negocio.	Anotación de las reglas del negocio que están involucradas con la UI.
Diagrama de navegación de ventanas	Permite declarar cuáles ventanas están disponibles y muestra las rutas de navegación posibles entre ellas.

Elemento	*Descripción*
Disposición de ventanas	Se realiza por cada ventana del diagrama de navegación, donde se muestra la manera en que ésta aparecerá ante el usuario.
Miniespecificación de ventana	Aspecto técnico que define el comportamiento para la apertura y cierre de la ventana y la activación y ejecución de cada botón, control y elemento de menú.
Especificación de campo	Define los campos y ediciones asociadas para todos los datos que aparecen en la ventana, que incluye una miniespecificación sobre la manera en que se adquieren los datos; además nombres de tablas, nombres de columnas, relaciones y criterios de selección.

4.7. Diseño de los procesos intranet

El diseño de procesos permite especificar y definir los aspectos algorítmicos internos de la lógica de negocios; es decir, el plano para el código de la aplicación. El diseño de procesos está influenciado por el paradigma y las capacidades de los lenguajes de destino que se emplearán para construir el sistema.

El diseño de los procesos intranet se soporta en base a tres aspectos: el diseño de arquitectura intranet, el diseño de la base de datos y la especificación de la interfaz. Se deberá ser capaz de hacer integraciones de varios ámbitos (tipos de lógicas) que comprenden accesos a bases de datos, elementos estructurados, elementos orientados a objetos y características de cuarta generación.

4.7.1. Lenguaje de definición de procesos intranet

Un Lenguaje de Definición de Procesos Intranet (LDPI) es un pseudocódigo que tiene las siguientes características:

- **Diccionario.** Conjunto de palabras clave.
- **Sintaxis.** Conjunto de reglas sintácticas para la construcción del lenguaje de marcación.
- **Lenguaje natural.** Para describir las características de procesamiento.

El objetivo de un diseño en LDPI es especificar la funcionalidad correspondiente, sin entrar en los aspectos detallados de cada uno de los lenguajes requeridos, por lo que el diseñador puede dejar a un lado los detalles que no son relevantes en el momento. Cuando el programador recibe el diseño en LDPI, es posible codificar en menor tiempo.

Ejemplo de LDPI

Un ejemplo de lenguaje de definición de procesos intranet es el siguiente:

```
<LDPI=HTML>
  <LDPI=JavaScript>
    usuario:focus
    usuario:valida nulo
    clave:valida nulo
  </LDPI>

  <LDPI=CGI>
    ⇒ Indicador de entrada
    inicial ← campoURL("x")
    if inicial = 0
      <LDPI=HTML>
      input hidden=1
      </LDPI>
    else
      <LDPI=HTML>
      input hidden=inicial
      </LDPI>
  </LDPI>

  input usuario
  input clave
  input aceptar:acción=verifacc.html
  input reset
</LDPI>
```

En el ejemplo anterior se muestran cinco definiciones, tres para HTML, una para JavaScript y otra para un CGI. El símbolo de "⇒" indica un comentario. Nótese que la definición de CGI indica particularmente un elemento dinámico, por lo tanto, el ejemplo de LDPI muestra una EW estática con datos dinámicos.

4.7.2. Interacción de lógicas

El ejemplo de la sección anterior muestra una definición sencilla en LDPI, sin embargo, el diseño de los procesos requiere una serie de definiciones mucho más elaboradas y que muchas veces no son muy fáciles de expresar, ya que implican sincronía entre una o varias EW. En este caso, una o varias definiciones en una EW pueden considerarse como un ámbito que opera bajo ciertas condiciones.

Cada uno de los ámbitos que se encuentran en una EW, corresponde a una lógica. Una lógica es un ámbito que describe funcionalidad y procesamiento. Las lógicas pueden ser de dos tipos:

- **Lógica controladora.** Es aquel ámbito empleado de una EW, que lleva el control de la funcionalidad de la misma. Puede mantener o ceder dicho control por medio de otra lógica de control.
- **Lógica declarativa.** Son aquellos ámbitos que realizan una función específica dentro de la EW pero no pueden alterar el control de la misma.

Toda lógica controladora puede ser una lógica declarativa, pero no toda lógica declarativa puede ser una lógica controladora. Para ayudar en la determinación y distinción de los tipos de lógicas, el planteamiento consiste en preguntar quién toma el control en la jerarquía más alta de decisiones, de acuerdo al ámbito presente.

La interacción entre los diferentes ámbitos permite que el control de la aplicación o del sistema fluya desde la interfaz, hasta la base de datos y de regreso, con intervención o no de la lógica del negocio. A lo largo del proceso, se distinguen varias propiedades entre las lógicas, las cuales se describen a continuación.

Características de la lógica controladora

- Permite especificar los aspectos dinámicos (datos y generación) de una EW, como son en la misma interfaz, en la lógica del negocio y en la base de datos.
- Puede estar incrustada dentro de otra lógica controladora.
- No necesariamente retoma el control, ya que otro ámbito interno al actual puede mandar el control a otra EW.

Características de la lógica declarativa

- Permite definir los aspectos estáticos de una EW.
- Establece las secciones de presentación e interacción con el usuario.
- Ejecuta tareas en su mismo ámbito especificado por el LDPI.

Ejemplo de lógicas

En el ejemplo en LDPI de la página 43, muestra cuatro ámbitos, a saber:

- El primer ámbito englobado por la primera declaración <LDPI=HTML> y que corresponde al ámbito principal.
- <LDPI=JavaScript> define el segundo ámbito con una instrucción propia de la interfaz.
- La interacción con un CGI se define por <LDPI=CGI> y pertenece al tercer ámbito.
- Finalmente, el cuarto ámbito tiene liberación en secuencias de HTML y está definido dentro de tercero.

4.8. Comentarios

La etapa de análisis para los SII enfatiza la obtención de interfaces primarias para que al pasar a la etapa del diseño pueda diseñarse la arquitectura del sistema y proceder con pasos de diseño más detallados. El resumen de los pasos del diseño para SII es el siguiente:

- De acuerdo al enfoque escogido obtener la especificación genérica de interfaces de usuario bajo un contexto web.

- Arquitectura.
  - ✓ Obtener el diagrama de hipervínculos.
- Base de datos.
  - ✓ Proceder con el diseño de forma tradicional.
  - ✓ En caso de utilizar el enfoque orientado a objetos, realizar correspondencias de clases y asociaciones con el modelo relacional.
- Interfaz de usuario.
  - ✓ Desarrollar un prototipo conceptual y evaluarlo con el usuario a través de las pruebas de uso.
  - ✓ Desarrollar un prototipo detallado y evaluarlo con el usuario a través de las pruebas de uso ajustadas.
- Procesos intranet.
  - ✓ De acuerdo a los elementos que integran la interfaz de usuario, establecer la funcionalidad de la interfaz en LDPI.
  - ✓ Establecer los ámbitos involucrados en la interfaz.
  - ✓ Realizar la interacción de lógicas para la integración de la lógica del negocio y consultas a la base de datos.

Capítulo 5. Aplicación de la Metodología: Un Caso de Estudio

Después de haber discutido los planteamientos de la metodología de diseño, el objetivo de este capítulo es formular un caso de estudio que permite aplicar e ilustrar el uso de la propuesta.

El caso de estudio presentado comprende varios módulos, de los cuales sólo será detallado uno de ellos para que el estudio sea manejable en el presente trabajo. Se enfatizará y detallará sobre los aspectos del diseño, dado que es el tema concerniente.

5.1. Presentación del problema

5.1.1. Requisitos

Una compañía de autobuses está dividida en dos grandes áreas:

- Gestión de pasajeros. La gestión de pasajeros se encarga de las reservaciones de asientos de autobús y dar información a los pasajeros y agencias de viajes sobre horarios y tarifas.
- Gestión de servicios. La gestión de servicios organiza los viajes ordinarios y especiales.

Se omite la descripción de las oficinas de contabilidad y de personal, así como de la venta de billetes, tampoco se considera la contabilidad de ventas después de los viajes. En particular, sólo se tratará la gestión de los pasajeros e información general de la compañía.

Información general de la compañía

La compañía presenta sus objetivos y la descripción de los servicios que presta a los clientes.

Gestión de pasajeros

Los clientes o agencias de viajes hacen reservaciones gratuitas de los asientos directamente en cualquier viaje ordinario de autobús. Se pueden hacer reservaciones por viaje completo o por segmentos de viaje, así que el mismo asiento puede estar disponible para diferentes

personas en diferentes partes del mismo viaje. Se dispone de consulta (cálculo) de tarifas y calendarización de viajes.

Todos los datos anteriores se registran en hojas de descripción de viaje, las cuales están disponibles con un mes de anticipación a la fecha del viaje. Se incluye el horario del viaje, lista de paradas intermedias y espacios para asiento, donde es posible anotar las reservaciones.

Se requiere contar con reservaciones por Internet para los habitantes de la Cd. de México, las cuales se podrán hacer hasta 24 horas antes de la salida y solamente en la terminal es posible hacerlas hasta 5 minutos antes de la salida. Las agencias de viaje deberán hacer reservaciones para los clientes que no tengan acceso a Internet.

Objetivos

- Proporcionar la información general de la empresa a través de Internet.
- Ofrecer servicio gratuito de reservaciones de autobuses a través de Internet.
- Proporcionar el servicio para los habitantes de la Cd. de México.
- Proporcionar información específica de los viajes.
- Realizar reportes de las hojas de descripciones de los viajes para el personal administrativo.

Vectores de calidad

- Facilidad de uso. Ambiente intuitivo, ya que los clientes no son expertos en el manejo de sistemas.
- Confiable. El sistema deberá proporcionar el servicio las 24 horas del día y responder eficientemente en los periodos vacacionales.
- Flexible. El sistema deberá garantizar que las transacciones se concluyan satisfactoriamente. Se ha determinado que cualquier proceso de reservación que se haga exactamente a las 24 horas antes de la salida del viaje puede terminar satisfactoriamente. Se ha estimado un promedio de 15 minutos para que el cliente seleccione y concrete sus opciones de viaje, y pueda registrar su reservación.

5.2. Análisis del sistema

5.2.1. Enfoque estructurado: diagrama de flujo de datos

El diagrama de flujo de datos para la gestión de pasajeros se muestra en la Figura 14, nótese que el diagrama se ilustra como un DFD de alcance expandido.

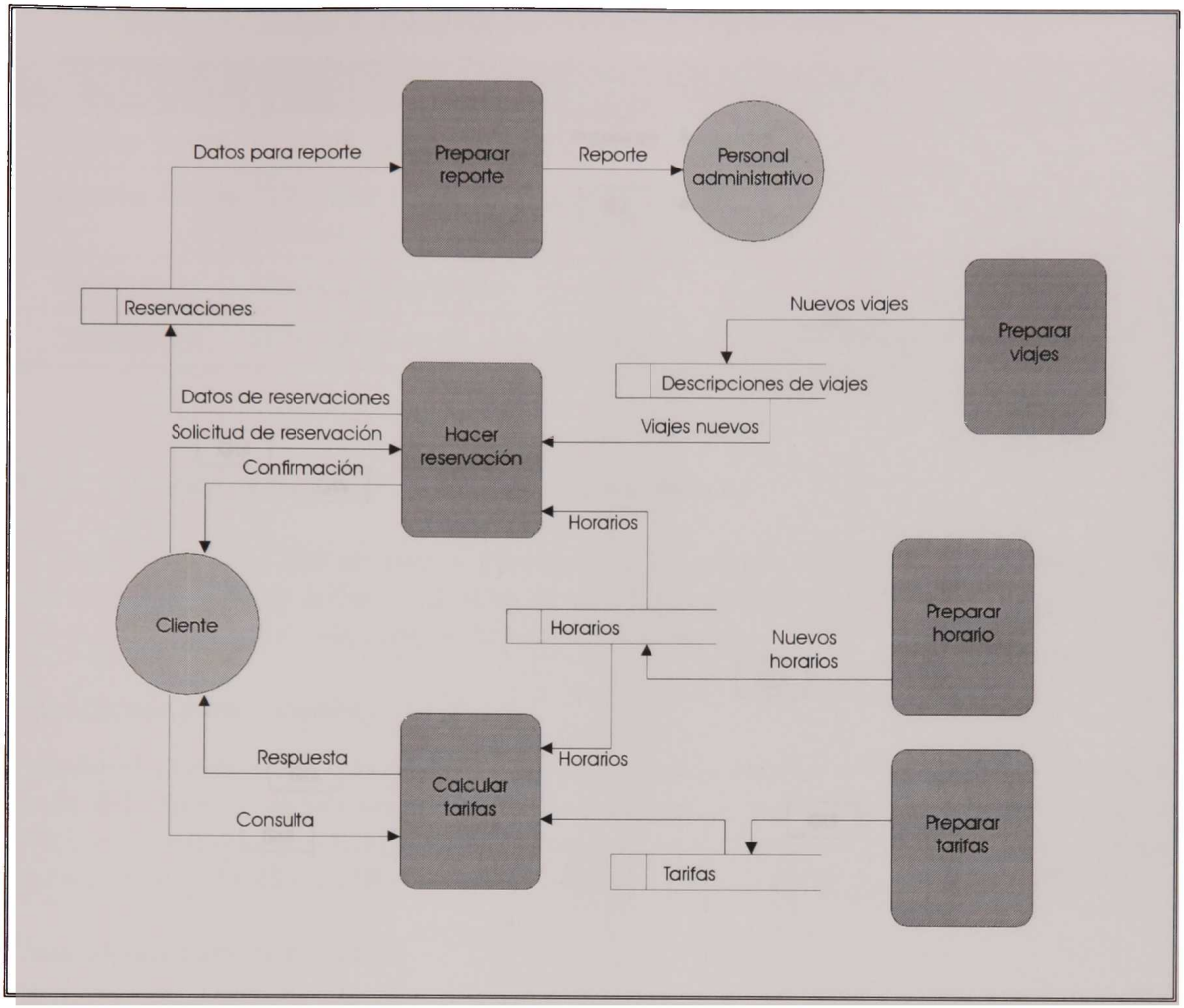


Figura 14. Diagrama de flujo de datos para gestión de pasajeros.

El diagrama de flujo de datos de la Figura 15 representa el mismo diagrama de flujo de datos de la Figura 14, pero ahora con las anotaciones de los mecanismos de transporte correspondientes.

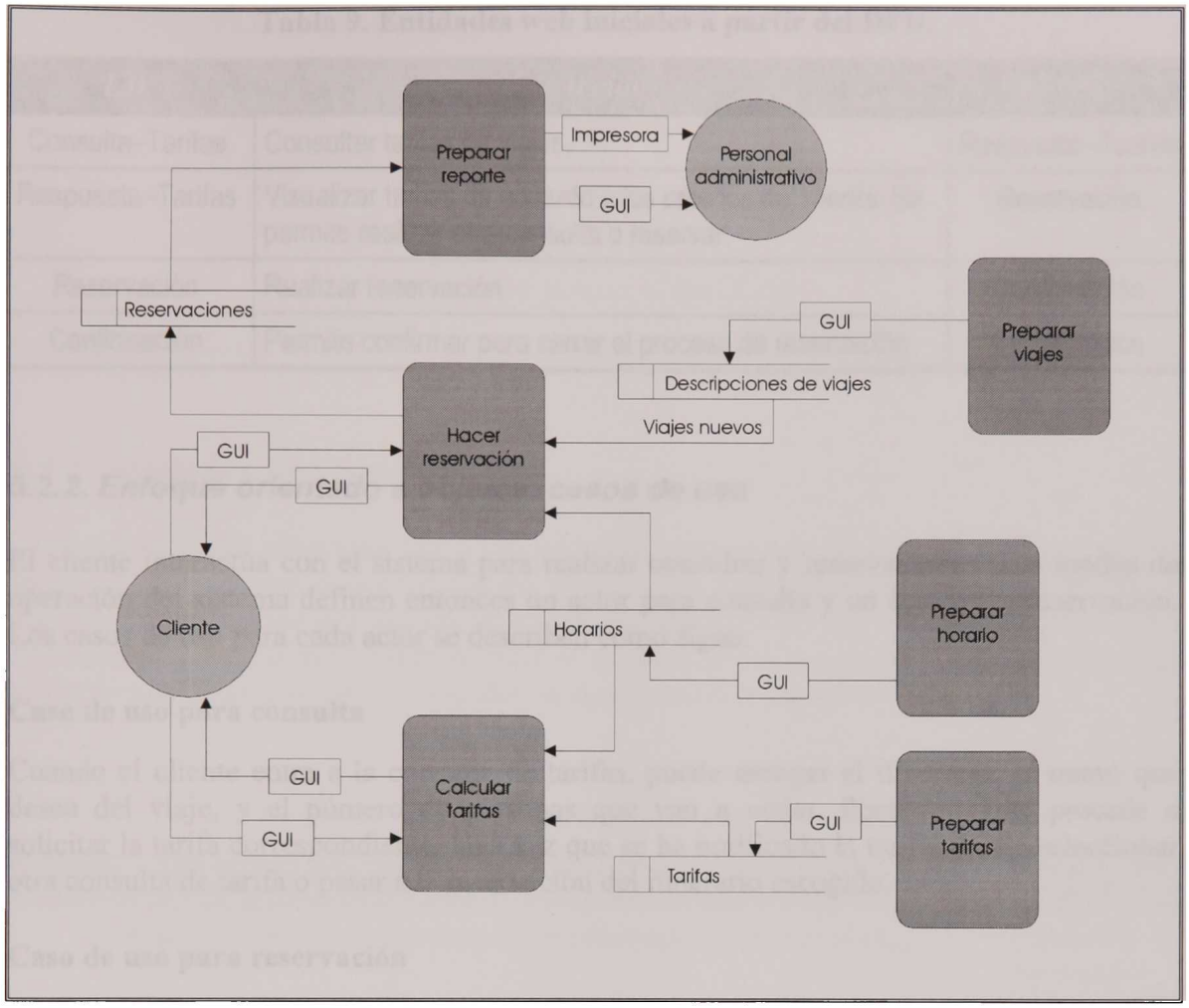


Figura 15. DFD con anotaciones de los mecanismos de transporte.

Los mecanismos de transporte en el DFD que representan interfaces con interacción del cliente se listan a continuación.

- GUI para consulta de tarifas.
- GUI para respuesta de tarifas.
- GUI para solicitud de reservación.
- GUI para confirmación de reservación.

Cada una de las GUI listadas anteriormente, permitirán representar las entidades web primarias (EW). Las EW con sus acciones y los vínculos correspondientes se listan en la siguiente tabla.

Tabla 9. Entidades web iniciales a partir del DFD.

EW	*Acción*	*Vínculo*
Consulta–Tarifas	Consultar tarifas de viajes.	Respuesta–Tarifas
Respuesta–Tarifas	Visualizar tarifas de acuerdo a los criterios del cliente. Se permite realizar otra consulta o reservar.	Reservación
Reservación	Realizar reservación	Confirmación
Confirmación	Permite confirmar para cerrar el proceso de reservación.	Confirmación

5.2.2. Enfoque orientado a objetos: casos de uso

El cliente interactúa con el sistema para realizar consultas y reservaciones. Los modos de operación del sistema definen entonces un actor para consulta y un actor para reservación. Los casos de uso para cada actor se describen como sigue.

Caso de uso para consulta

Cuando el cliente entra a la consulta de tarifas, puede escoger el itinerario, el tramo que desea del viaje, y el número de personas que van a viajar. Posteriormente procede a solicitar la tarifa correspondiente. Una vez que se ha notificado la tarifa, puede seleccionar otra consulta de tarifa o pasar a la reservación del itinerario escogido.

Caso de uso para reservación

Al entrar el cliente a la opción de reservación se le muestra el itinerario escogido, se le piden sus datos personales, fecha y horario deseado. Una vez que haya proporcionado los datos necesarios, se notifica al cliente que confirme. La confirmación la efectúa el cliente con la revisión de sus datos.

Entidades web iniciales

Al analizar el caso de uso para consulta, la primera frase ("el cliente entra a la consulta de tarifas") permite determinar una entidad la cual se llamará Consulta–Tarifa. La segunda frase ("solicitar la tarifa"), indica la acción correspondiente que efectúa Consulta–Tarifa. La frase siguiente ("se ha notificado la tarifa") establece otra entidad que notifica un resultado, en este caso Notifica–Tarifa. En esa misma frase se tiene un par de acciones ("puede seleccionar otra consulta de tarifa o pasar a la reservación").

De la misma forma, el caso de uso para reservación define una entidad para reservar, Reservación, cuya acción está implícita. Como que la notificación consiste en la revisión de sus datos, se tiene otra acción. Por lo tanto, las entidades web iniciales se muestran en la siguiente tabla.

Tabla 10. Entidades web iniciales a partir de casos de uso.

EW	*Acción*	*Vínculo*
Consulta–Tarifa	Solicita una consulta de tarifa.	Notifica–Tarifa
Notifica–Tarifa	Selección de otra consulta. En su caso, pasar a la reservación.	Consulta–Tarifa/ Reservación
Reservación	Realizar la reservación correspondiente y pedir confirmación.	Reservación

5.2.3. Entidades web revisadas

Las entidades web obtenidas deben ser revisadas nuevamente de acuerdo a los requisitos, separando las acciones implícitas en cada una de ellas. Esto permite obtener otras entidades que deben listarse con otra acción, posiblemente derivada de una acción más general. La especificación bajo contexto web de las EW iniciales, junto con aquellas que corresponden a la información general de la compañía, se muestran en la siguiente tabla.

Tabla 11. Entidades web iniciales revisadas.

EW	*Acción*	*Vínculo*
Presenta	Presentar opciones sobre información y reservaciones.	Objetivos/ Servicios/ Consulta–Tarifas
Objetivos	Mostrar objetivos de la compañía.	Presenta
Servicios	Mostrar servicios que ofrece la compañía.	Presenta
Consulta–Tarifas	Consultar tarifas de viajes.	Muestra–Tarifas
Muestra–Tarifas	Visualizar tarifas de acuerdo a los criterios del cliente. Permitir realizar otra consulta.	Consulta–Tarifas
Muestra–Tarifas	Continuar con la reservación.	Reserva
Reserva	Realizar reservación	Confirma
Confirma	Confirmar para cerrar el proceso de reservación.	Confirma
Confirma	Presentar nuevamente opciones iniciales.	Presenta

5.3. Diseño de la arquitectura del sistema

Cada uno de los renglones de la especificación de entidades, se traduce en las correspondientes representaciones de un diagrama H. De acuerdo con la Tabla 11, el diagrama H obtenido se muestra en la siguiente figura.

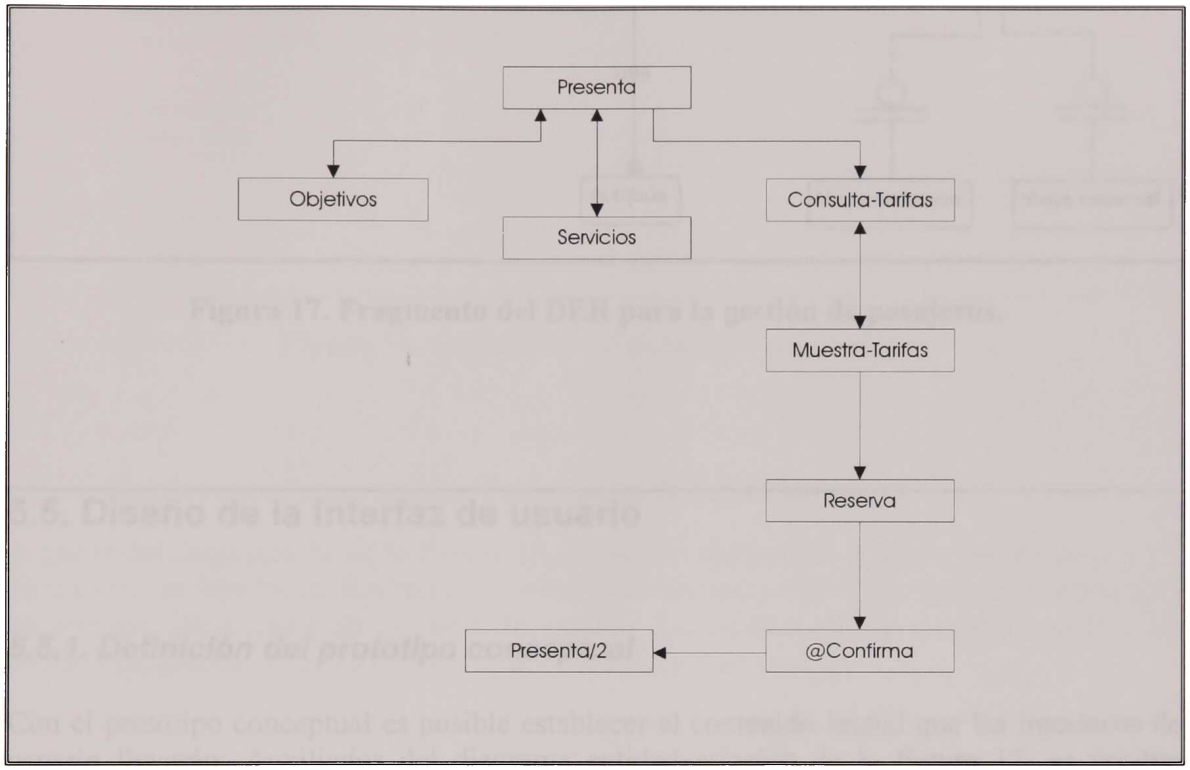


Figura 16. Diagrama H para la gestión de pasajeros e información de la compañía.

Como se observa, el diagrama H obtenido tiene una representación estructurada, con una organización lineal.

5.4. Diseño de la base de datos

Como se especificó anteriormente, el diseño de la base de datos sigue el enfoque relacional, y los elementos de entrada junto con las correspondencias necesarias para la orientación a objetos no se discutirán. Solo se presentará un fragmento del diagrama entidad–relación que servirá como parte de la definición del contenido de las interfaces de usuario.

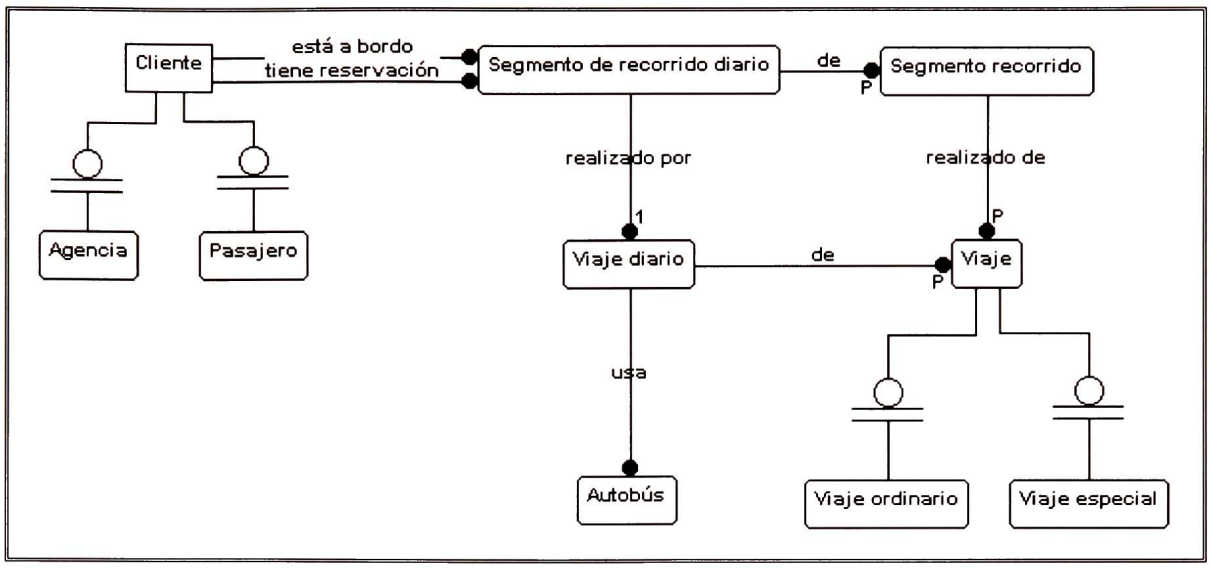


Figura 17. Fragmento del DER para la gestión de pasajeros.

5.5. Diseño de la interfaz de usuario

5.5.1. Definición del prototipo conceptual

Con el prototipo conceptual es posible establecer el contenido inicial que las interfaces de usuario llevarán: Auxiliados del diagrama entidad–relación de la Figura 17, es posible determinar el contenido necesario. Para esto, la relación

Cliente – tiene reserva – Segmento de recorrido diario

permite definir que el Cliente debe tener una descripción detallada del o los viajes que tiene reservados. Entonces, el Cliente puede tener varios Segmentos de recorrido, y a cada Segmento de recorrido le corresponde al menos un viaje del Cliente. Así, el Cliente tiene sus datos y una serie de diferentes recorridos que ha observado, por lo que el prototipo conceptual toma la forma mostrada en la Figura 18.

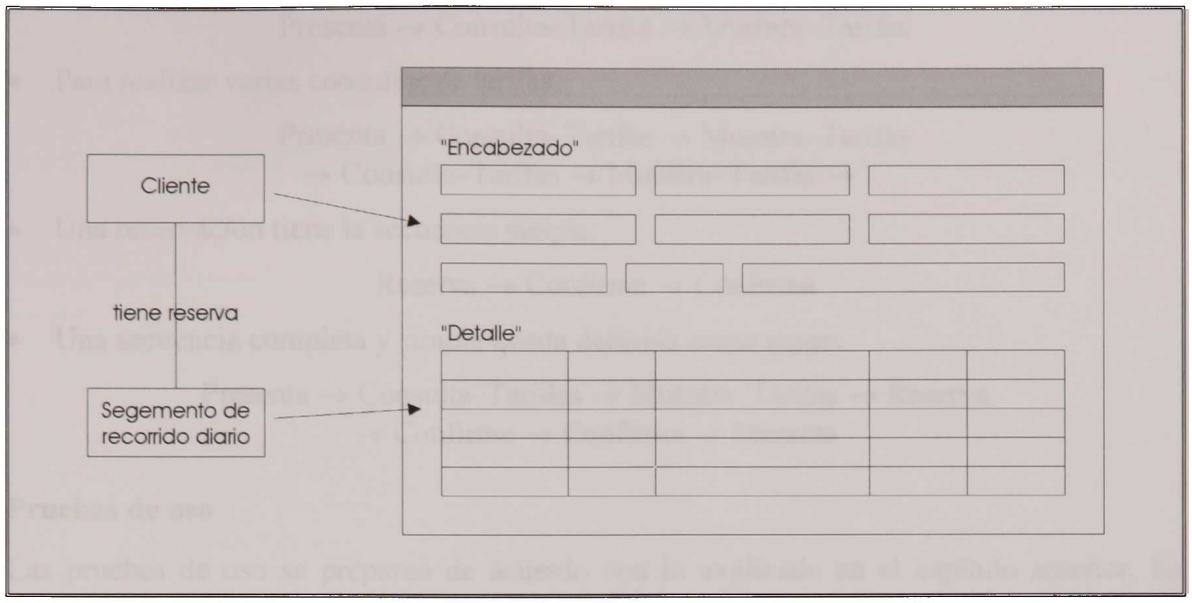


Figura 18. Contenido del prototipo conceptual.

Obtención de rutas de navegación y secuencias de interfaces

A partir del diagrama H de la Figura 16 se pueden derivar las rutas de navegación y las secuencias de interfaces. Sin embargo, rutas y secuencias pueden ser ajustadas con la lista de eventos proveniente de la etapa de análisis. La siguiente tabla muestra una lista de eventos ordenados cronológicamente para consulta y reservación.

Tabla 12. Eventos ordenados cronológicamente.

ID	*Evento*	*Tipo*
1	El cliente entra a la página principal de la empresa.	I
40	El cliente consulta.	I
130	El cliente selecciona un viaje ordinario.	E
50	El cliente hace una reservación.	I

Los tipos de eventos mostrados en la tabla anterior son (I)= inesperados y (E)=esperados. El orden de los eventos es coincidente con las representaciones que se tiene en el diagrama H, tanto para los hipervínculos, como para las EW relacionadas. El mismo diagrama H define entonces las rutas de navegación para la gestión de pasajeros. Las secuencias de interfaces que se pueden obtener son las siguientes:

- Para realizar una simple consulta de tarifas:

Presenta → Consulta–Tarifas → Muestra–Tarifas

- Para realizar varias consultas de tarifas:

Presenta → Consulta–Tarifas → Muestra–Tarifas
→ Consulta–Tarifas → Muestra–Tarifas → ...

- Una reservación tiene la secuencia simple:

Reserva → Confirma → Confirma

- Una secuencia completa y simple queda definida como sigue:

Presenta → Consulta–Taridas → Muestra–Tarifas → Reserva
→ Confirma → Confirma → Presenta

Pruebas de uso

Las pruebas de uso se preparan de acuerdo con lo explicado en el capítulo anterior. Es necesario que el usuario final revise y apruebe los prototipos conceptuales de interfaz y vea si las disposiciones de navegación son las correctas y también las apruebe.

El prototipo conceptual se recomienda utilizarlo de una a dos veces, y una vez aprobados los prototipos conceptuales y las navegaciones será posible pasar a prototipos detallados (alta tecnología), con el mínimo riesgo de cambios grandes.

5.5.2. Prototipo detallado

El prototipo detallado se construye directamente a partir del prototipo conceptual aprobado por el usuario. Una vez que se ha construido, es necesario también revisarlo con el usuario para que lo apruebe.

Es importante resaltar que el prototipo detallado se transformará en la interfaz del usuario, por lo que antes de implementar la lógica de negocios es necesario proveer de la funcionalidad necesaria a las interfaces. Una vez que el prototipo detallado se ha aprobado como interfaz de usuario, es necesario documentar la interfaz y el módulo correspondiente.

La Figura 19 muestra un prototipo detallado que se aprueba como interfaz de usuario. La Tabla 13 muestra el contenido de la documentación para el módulo correspondiente, y la Tabla 14 muestra la documentación para la interfaz de usuario.

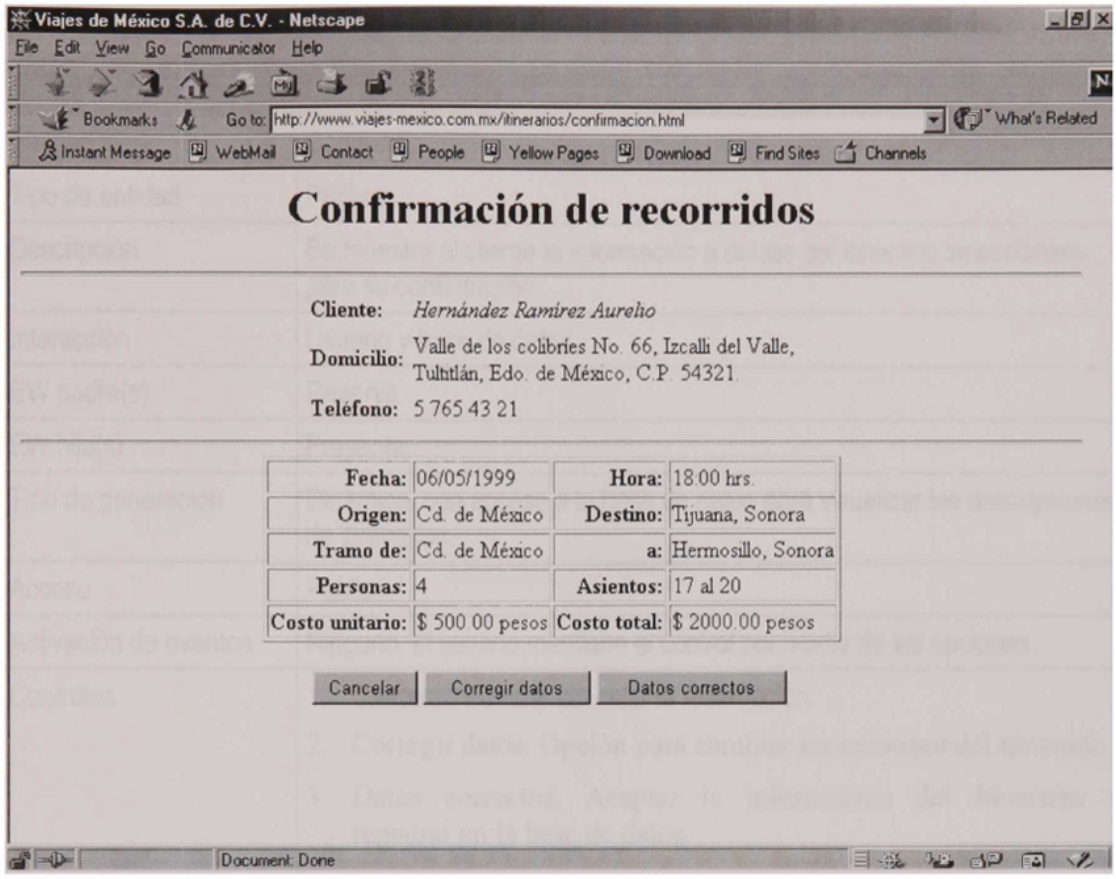


Figura 19. Prototipo detallado para la EW Confirmación.

Tabla 13. Contenido de la documentación para módulo.

Elemento	*Descripción*
Módulo	Reservaciones.
Descripción	El módulo de reservaciones permite al usuario consultar tarifas a partir de las opciones de itinerarios y efectuar reservaciones.
Comunicación	Módulo "Información".
EW involucradas	Consulta–Tarifas, Muestra–Tarifas, Reserva, Confirma.
Observaciones	Ninguna.

Tabla 14. Contenido de la documentación para interfaz de usuario.

Elemento	*Descripción*
Entidad web	Confirma.
Tipo de entidad	Pública.
Descripción	Se muestra al cliente la información a detalle del itinerario seleccionado, para su confirmación.
Interacción	Usuario y base de datos.
EW padre(s)	Reserva.
EW hija(s)	Presenta.
Tipo de generación	Dinámica, con acceso a la base de datos para visualizar las descripciones de itinerarios.
Acceso	Público.
Activación de eventos	Ninguno, el usuario mantiene el control por medio de las opciones.
Controles	1. Cancelar. Permite cancelar la reservación. 2. Corregir datos. Opción para cambiar las opciones del itinerario. 3. Datos correctos. Aceptar la información del itinerario y registrar en la base de datos.
Lógica del negocio	Comunicación con le módulo de "Validación de itinerarios".
Validaciones	Ninguna.
Observaciones	Ninguna.

5.6. El diseño de procesos intranet

Se mencionó que el diseño de los procesos intranet está influenciado por los paradigmas de desarrollo empleados. Sin embargo, hay que hacer integraciones con los procesos de base de datos, estructurados, de objetos y de cuarta generación. Para ejemplificar la integración adecuada de todos estos elementos, se desarrollará el diseño de acceso al sistema para los ejecutivos de la compañía, el cual simplemente pide un usuario y una clave de acceso.

Se ha determinado que para acceder al sistema, es necesario validar el usuario y la clave de acceso con la base de datos. Si los datos existen, se puede acceder al sistema, de lo contrario se notificará con una aviso, y se le solicita al usuario nuevamente sus datos. Este proceso solo debe efectuarse un máximo de tres intentos. El prototipo detallado se muestra en la Figura 20 y en la Tabla 15 se detalla la especificación correspondiente.

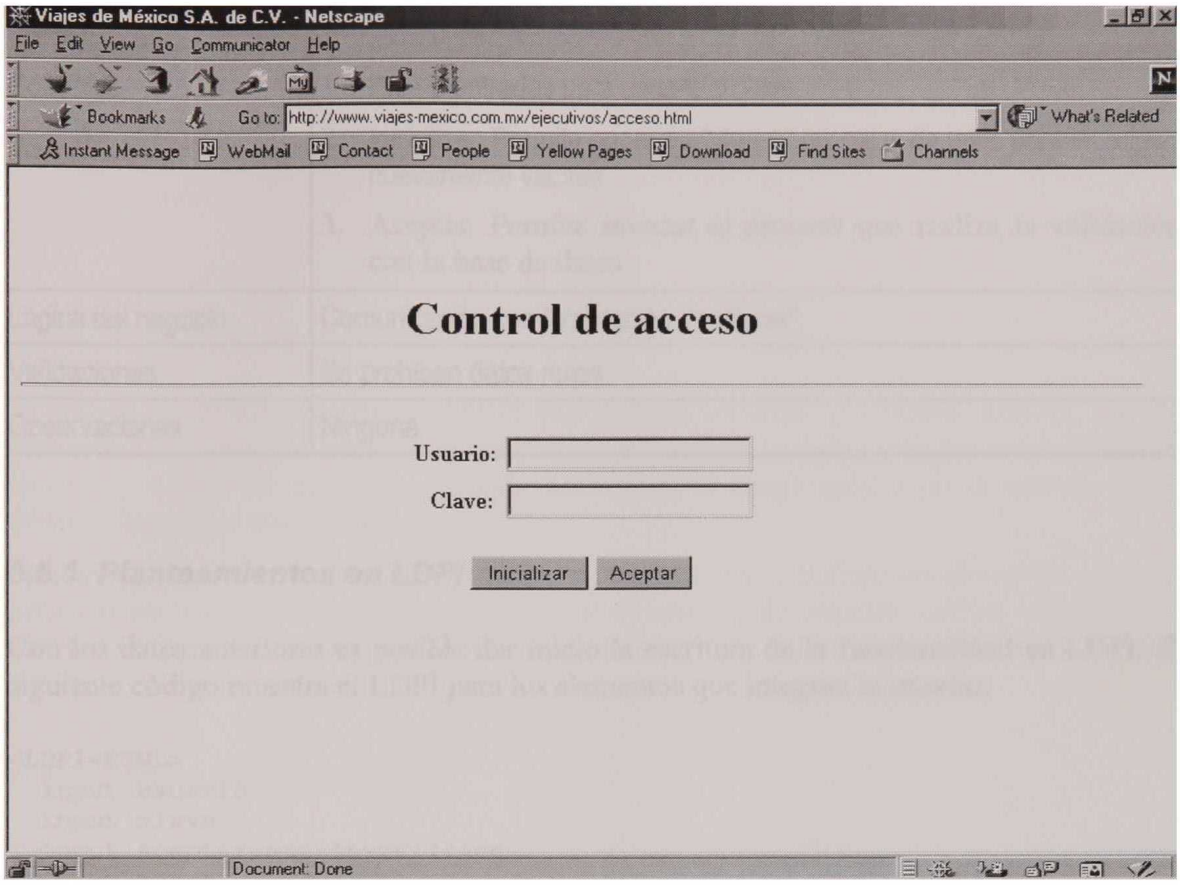


Figura 20. Prototipo detallado para acceso al sistema.

Tabla 15. Especificación de interfaz para acceso al sistema.

Elemento	*Descripción*
Entidad web	Acceso.
Tipo de entidad	Privada.
Descripción	Se solicitan los datos de usuario y clave para poder accesar al sistema.
Interacción	Usuario y base de datos.
EW padre(s)	Principal.
EW hija(s)	Opciones.
Tipo de generación	Estática.
Acceso	Privado.
Activación de eventos	Enfocar el campo de usuario desde el inicio.

Elemento	*Descripción*
Controles	1. Par de entradas para usuario y clave. 2. Inicializar. Permite poner en "blanco" las cajas de texto para introducir nuevamente valores. 3. Aceptar. Permite invocar al proceso que realiza la validación con la base de datos.
Lógica del negocio	Comunicación con "Validación de claves".
Validaciones	Se prohiben datos nulos.
Observaciones	Ninguna.

5.6.1. Planteamientos en LDPI

Con los datos anteriores es posible dar inicio la escritura de la funcionalidad en LDPI. El siguiente código muestra el LDPI para los elementos que integran la interfaz.

```
<LDPI=HTML>
  input usuario
  input clave
  input aceptar:acción=verifacc
  input reset
</LDPI>
```

Nótese que hay una acción para "aceptar", esto obedece a la comunicación que se necesita con el módulo de "Validación de claves". Ahora, se requiere enfocar el campo de usuario desde el inicio, lo cual se puede expresar de la siguiente manera.

```
  <LDPI=JavaScript>
    usuario:focus
  </LDPI>
```

Para evitar los datos nulos, se requiere validar el contenido de las cajas de texto, para esto se puede expresar:

```
<LDPI=JavaScript>
    usuario:valida nulo
    clave:valida nulo
</LDPI>
```

Los fragmentos de LDPI anteriores definen los controles y permiten establecer la funcionalidad de la interfaz, el código resultante es el siguiente.

```
<LDPI=HTML>
  <LDPI=JavaScript>
    usuario:focus
```

```
    usuario:valida nulo
    clave:valida nulo
  </LDPI>
  input usuario
  input clave
  input aceptar:acción=verifacc
  input reset
</LDPI>
```

5.6.2. Definición de ámbitos

El código resultante permite establecer dos ámbitos iniciales, el primero definido por los elementos (inputs) que integran la interfaz, y el segundo se define por las validaciones de usuario y clave. De esta forma queda establecida la funcionalidad de la interfaz y los ámbitos que están involucrados.

También, el primer ámbito define una lógica declarativa al definir los elementos que son estáticos en la interfaz y que interactúan con el usuario. El segundo ámbito está definido como una lógica de control al efectuar operaciones sobre los elementos estáticos de la interfaz.

5.6.3. Lógica del negocio

Ahora, la lógica del negocio "Validación de claves" es un simple proceso el cual verifica que usuario y clave existan en la base de datos. Si existen en la base de datos se permite el acceso al sistema de acuerdo al nivel correspondiente, de lo contrario se notificará que las claves son incorrectas. Validación de las claves ha quedado especificado bajo una acción que corresponde al input aceptar del código obtenido y responde al nombre de Verifacc.

El planteamiento inicial de Verifacc en LDPI es como sigue:

```
<LDPI=CGI>
    if cuanta <= 3 do
      encontrar usr, cve en la BD
      if existen
        Accesar el sistema
      else
        Claves incorrectas
    end
    else do
      Desactivar sistema
      Terminar
    end
</LDPI>
```

Nótese que se tiene un ámbito distinto, especificado por <LDPI=CGI>. El planteamiento anterior de LDPI debe ser refinado a partir de las especificaciones de procesos, en caso de utilizar el enfoque estructurado, o en su caso, a partir de las especificaciones de responsabilidades de un modelo CRC.

5.6.4. LDPI finales

Antes de presentar los LDPI finales, es necesario considerar un aspecto más, el número de intentos para accesar al sistema. Para esto se necesita al menos una variable que almacene el valor de dichos intentos cuando el usuario introduce sus datos.

```
<LDPI=CGI>
  ⇒ Indicador de entrada
  inicial ← campoURL("x")
</LDPI>
```

LDPI para proggen.html

```
<LDPI=HTML>
  <LDPI=JavaScript>
    usuario:focus
    usuario:valida_nulo
    clave:valida_nulo
  </LDPI>

  <LDPI=CGI>
    ⇒ Indicador de entrada
    inicial ← campoURL("x")
    if inicial = 0
      <LDPI=HTML>
      input hidden=1
      </LDPI>
    else
      <LDPI=HTML>
      input hidden=inicial
      </LDPI>
  </LDPI>

  input usuario
  input clave
  input aceptar:acción=verifacc.html
  input reset
</LDPI>
```

LDPI para verifacc.html

```
<LDPI=HTML>
  <LDPI=CGI>
    usr ←campoURL("usuario")
    cve ←campoURL("clave")
    cuenta ←campoURL("oculto")

    if cuanta <= 3
      encontrar usr, cve en la BD
      if existen
        <LDPI=HTML>
```

```
          Accesar el sistema
          <LDPI=Script>
          submit según nivel
          </LDPI>
        </LDPI>
      else
        <LDPI=HTML>
          Claves incorrectas
          <LDPI=Script>
          submit proggen.html
          </LDPI>
        </LDPI>

    else
      <LDPI="HTML">
        Desactivar sistema
        <LDPI=Script>
        Terminar
        </LDPI>
      </LDPI>
  </LDPI>
</LDPI>
```

5.7. Construcción de procesos

5.7.1. Ejemplos de código

Las siguientes líneas de código muestran la traducción de los LDPI planteados en la sección anterior y en este caso corresponden a códigos de WebSpeed. El primer código corresponde a la interfaz que pide usuario y clave; mientras que el segundo código se encarga de todas las validaciones correspondientes en la base de datos.

Código 1: proggen.html

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta name="AUTHOR" content="Ulises Juarez Martinez">
<title>Viajes de México S.A. de C.V.</title>
</head>

<body bgColor="white" onLoad="document.acceso.txtUsr.focus()">
<center>
<p> </p><p> </p>
<font size=+4 color="blue">Control de acceso</font>
<p> </p><p> </p>
<form name="acceso" method="post" action="verifacc.html">
```

```
<script language="SpeedScript">

  DEF VAR cuenta AS INT.

  ASSIGN cuenta = INTEGER(get-field("num")).

  IF cuenta = 0 THEN DO:
</script>

<input type="hidden" name="num" value="1">

<script language="SpeedScript">
  END.
  ELSE DO:
</script>

<input type="hidden" name="num" value="`cuenta`">

<script language="SpeedScript">
  END.
</script>

<table border=5>
  <tr>
    <td align=left><b>Usuario:</b></td>
    <td><input type="text" name="txtUsr" size="11" value=""
               onFocus="this.select()"
               onChange="this.value=this.value.toUpperCase()"></td>
  </tr>
  <tr>
    <td align=left><b>Contrase&ntilde;a:</b></td>
    <td><input type="password" name="pswUsr" size="3" value=""
               onFocus="this.select()"></td>
  </tr>
</table>
<p><p>
<input type="submit" name="sbmDatos" value="Aceptar">
<input type="reset" name="rstForma" value="Inicializar"
       onClick="document.acceso.txtUsr.focus()">
</form>
</center>

</body>
</html>
```

Código 2: verifacc.html

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta name="AUTOR" content="Ulises Juarez Martinez">
<title> Viajes de México S.A. de C.V.</title>
</head>
```

```
<body bgColor="white">
<script language="SpeedScript">

  DEF VAR usr AS CHAR.
  DEF VAR cve AS CHAR.
  DEF VAR cuenta AS INT.
  DEF VAR nivel AS CHAR FORMAT "X(12)" EXTENT 6 INIT
    ["niv01.html",
     "niv02.html",
     "niv03.html",
     "niv04.html",
     "niv05.html",
     "niv06.html"].

  ASSIGN usr = STRING(get-field("txtUsr"))
         cve = STRING(get-field("pswUsr"))
         cuenta = INTEGER(get-field("num")).

  IF cuenta <= 3 THEN DO:
    FIND meacceso WHERE meacceso.cveusr = usr AND
                        meacceso.cvealf = cve NO-ERROR.
    IF AVAILABLE meacceso THEN DO:
</script>

<p> </p><p> </p><p> </p>
<h1 align=center>ACCESANDO AL SISTEMA ...</h1>
<form name="control" method="post" action="`nivel[meacceso.nivel]`">
<input type="hidden" name="v_usr" value="`usr`">
</form>
<script language="JavaScript">
document.control.submit()
</script>

<script language="SpeedScript">
    END.
    ELSE
      ASSIGN cuenta = cuenta + 1.
  END.
  IF cuenta > 3 THEN DO:
</script>

<p> </p><p> </p><p> </p>
<h1 align=center>TRES ACCESOS NEGADOS POR EL SISTEMA<br>
                 DESACTIVANDO SISTEMA ...</h1>
<script language="JavaScript">
window.location = "http://home.netscape.com"
</script>

<script language="SpeedScript">
  END.
  ELSE DO:
    FIND meacceso WHERE meacceso.cveusr = usr AND
                        meacceso.cvealf = cve NO-ERROR.
    IF NOT AVAILABLE meacceso THEN DO:
</script>

<p> </p><p> </p><p> </p>
```

```
<h1 align=center>CLAVES INCORRECTAS<br>ACCESO NEGADO ...</h1>
<form name="retorno" method="post" action="proggen.html">
<input type="hidden" name="num" value="`cuenta`">
</form>
<script language="JavaScript">
document.retorno.submit()
</script>

<script language="SpeedScript">
    END.
  END.
</script>

</body>
</html>
```

Los dos códigos anteriores interactúan y sincronizan el control de acceso al sistema de la siguiente manera:

- Se recibe el usuario y contraseña (proggen.html), los cuales se envían al segundo código (verifacc.html) y se validan con la base de datos.
- Si los valores de usuario y contraseña no se encuentran en la base de datos, se regresa a proggen.html.
- El ciclo se establece tres veces. Si no hay correspondencia con los datos de la base después de los tres intentos, se llama a un URL determinado, y si hay correspondencia se llama a cualquiera de los archivos especificados, los cuales corresponden al nivel de usuario correspondiente.

5.8. Consideraciones para la construcción de los SII

Finalmente, esta sección considera algunos de los aspectos dentro de la construcción de los Sistemas de Información Intranet. Todos los aspectos están basados en las experiencias obtenidas al desarrollar varios sistemas, tanto para determinar la solución a la problemática como en la aplicación posterior de la metodología.

1. Las herramientas para generación de páginas HTML ayudan bastante cuando hay muchos elementos de formulario y/o cuando los formularios se tienen que hacer coincidir con los registros de la base de datos.
   - ✓ Si se tiene un mínimo de objetos, tales herramientas pueden ayudar sólo bajo casos aislados.
   - ✓ Cuando se requiere anexar elementos, posteriores al diseño, es recomendable realizarlo de forma manual, ya que habrá más libertad de hacerlo dada la abundante información de lógica de la herramienta.
2. La construcción de la interfaz de usuario debe seguir un orden adecuado. Una vez que se obtienen los objetos básicos y propios de la interfaz (del prototipo detallado) el desarrollo se recomienda que sea como sigue:

- ✓ Implementar la funcionalidad obtenida a partir de las pruebas de uso y los prototipos de tecnología básica.
- ✓ De ser necesario, implementar los aspectos avanzados para que la interfaz esté preparada para comunicarse con los componentes del negocio necesarios.
- ✓ La secuencia clásica de construcción obedece a un seguimiento como el siguiente: Interfaz (HTML) → Funcionalidad (JavaScript) → Lógica del negocio.

3. Cuando se tienen valores de iniciación en un frame, es necesario enviarlos después de que han tomado un valor. Esto lleva a dos planteamientos:
   - ✓ Separar la lógica en archivos de procedimientos para que cada frame los utilice y se evite la duplicidad de código.
   - ✓ Inicializar la lógica desde el frameset y enviar los valores a los frames que los utilicen.
4. Al establecer la comunicación entre páginas, es necesario especificar si los valores que se pasan entre las páginas pueden o no ser visibles por el usuario.
5. Cuando se requiere validar entradas específicas en los campos de texto, se puede optar por ventanas emergentes. Hay que considerar que la lógica de la interfaz puede complicarse.
6. Si hay que actualizar con la especificación de queries, es necesario dividir el diseño en dos EW, una para la captura de datos, y otra para la ejecución de los queries y mensajes adicionales (tal como en el ejemplo presentado en LDPI). Si se tiene la suficiente habilidad, se puede hacer una sola EW condicionada y darle dos o más salidas según sea necesario. Esto equivale a una secuencia de interfaces, pero todas son producto de un solo archivo.
7. Si el sistema requiere una dinámica de ventanas emergentes, la complejidad de los scripts (lógicas) puede ser muy alta, debido a la posibilidad de la coexistencia de scripts.
8. La simulación de páneles de navegación requiere de una alta retroalimentación de variables, la cual lleva a un manejo directo de punteros a los registros de la base de datos.
9. El tipo de fuentes usadas deben ser los que el sistema proporciona de forma preestablecida, ya que no todas las plataformas soportan las mismas fuentes. Incluso en una misma plataforma no se dispone de los mismos recursos.
10. Un aspecto importante es ocultar el código empleado en los scripts, lo cual se puede hacer al colocar el código de los scripts fuera del de los archivos HTML y en un directorio sin derecho de acceso a los usuarios.
11. La determinación de la lógica controladora es en ocasiones difícil, según:
    - ✓ La complejidad del problema.
    - ✓ El tipo de solución planteada.
    - ✓ La conjunción de tecnologías como HTML, JavaScript, y el lenguaje de la herramienta empleada.

- ✓ El grado de autoalimentación de variables en la misma página.
- ✓ El apego a los estándares, como en el caso del HTML.
- ✓ La claridad para definir los alcances y limitaciones entre la comunicación de lenguajes.
- ✓ El anidamiento de los scripts.

12. El flujo de la lógica controladora puede distribuirse en varias páginas para su mayor facilidad de implantación, con lo que se establece un ciclo que cierra las condiciones hasta que se cumpla alguna que permita seguir la secuencia o flujo del sistema.
13. Según la cantidad de elementos que integren una llave primaria, y al mismo tiempo, la cantidad de ellos presentes en una página u objeto, es la rapidez con que se podrá armar una llave para acceder un registro en una base de datos. Esto también reflejará la cantidad de páginas que se utilizarán y el grado de autollamadas a cada una de dichas páginas.
14. Todos los procesos de diseño son independientes de la plataforma de desarrollo.

Capítulo 6. Conclusiones

6.1. Metodología de diseño para SII

El objetivo principal del trabajo fue el desarrollar una metodología para el diseño de SII a partir de proyectos de desarrollo reales, lo cual hace que la propuesta tenga dos aspectos importantes:

1. Un fuerte fundamento práctico, es decir, incorpora parte de los procesos y métodos de desarrollo retroalimentados que los diseñadores utilizan para proyectos de SII.
2. Incorporación de bases sólidas de ingeniería de software a la naturaleza de la Web, considerando los dos enfoques más empleados para sistemas de información: el enfoque estructurado y el enfoque orientado a objetos.

6.1.1. Integración de la metodología con los enfoques estructurado y orientado a objetos

Otro de los objetivos fue el determinar el grado de aportación de los enfoques estructurados y orientados a objetos a la metodología propuesta. En realidad no es posible decir la aportación que los enfoques tienen al respecto, en tal caso, es ver la forma en cómo la metodología propuesta complementa dichos enfoques para incorporar la naturaleza de la Web. En este caso:

- Enfoque estructurado. La definición de los procesos derivados de un DFD se incorpora directamente como definiciones de procesos intranet.
- Enfoque orientado a objetos. Cada una de las clases obtenidas durante el análisis con el modelo CRC puede integrarse en una EW y el modelo Objeto–Relación define los mensajes entre las EW involucradas.

De esta forma, la metodología propuesta tiene compatibilidad e integración total con los enfoques estructurado y orientado a objetos, ya que todo el planteamiento surge a partir del enfoque seleccionado para desarrollar el sistema. Con esta perspectiva, el lenguaje de modelado unificado (UML) puede utilizarse sin ningún problema dentro de la metodología de diseño propuesta, ya que UML es un lenguaje de modelado (que va más allá de los alcances del software), y es independiente del proceso empleado.

6.1.2. Aplicación de la metodología

La metodología se ha aplicado a varios proyectos, de los cuales tres se han concluido y corresponden a desarrollos de SII, y actualmente se desarrollan otros proyectos, uno de los cuales es un desarrollo de páginas Web. Al aplicar la metodología en ambos tipos de proyectos, los diseños de arquitectura del sistema, de interfaz de usuario y de procesos intranet han dado resultados exitosos, aún para personas con poca o ninguna familiaridad con procesos de ingeniería de software.

6.1.3. Oportunidades y deficiencias

De lo anterior se pueden establecer algunas oportunidades y deficiencias que la metodología presenta.

Oportunidades

- Permite integrar la naturaleza web en sistemas de información.
- Es posible desarrollar los SII con enfoques estructurados u orientados a objetos.
- Es posible integrar métodos robustos como UML.
- Permite desarrollar páginas web sin infraestructura de sistema de información.

Deficiencias

- La integración con el enfoque orientado a objetos puede ser aún "burda".

6.2. Diseño de la arquitectura del sistema

El diseño de la arquitectura del sistema permite obtener un esqueleto estructurado y jerárquico de las entidades web involucradas, la cual es posible representar a través de los diagramas H.

6.2.1. Diagramas H

Los diagramas de hipervínculos mostraron ser una herramienta que permite representar en forma compacta varios aspectos de un sistema:

- Comunicación entre entidades web.
- Especificación de rutas de navegación.
- Define secuencias de interfaces.
- Representación jerárquica de los módulos que integran el sistema.
- Soportan la representación de arquitecturas para Intranets Extendidas (Extranets).

- Permite representar no solo a las entidades web que integran el sistema de información, sino también las entidades web estáticas que están caracterizadas por mostrar contenido específico.

6.3. Diseño de la base de datos

El diseño de una base de datos relacional para los sistemas de información intranet no requiere condiciones especiales o específicas. Por tal razón, el enfoque relacional, las técnicas para la distribución de datos, y las correspondencias con un enfoque orientado a objetos se aplican de la misma forma que en un sistema cliente/servidor.

6.4. Diseño de la interfaz de usuario

La complejidad que muestran los sistemas en general hace necesaria la utilización de prototipos no solo en la etapa de diseño, sino también en la etapa de análisis. Los prototipos conceptuales son herramientas que ayudan a plasmar las necesidades de los usuarios y las reglas del negocio en las etapas de análisis y diseño, respectivamente.

6.4.1. Prototipo conceptual

El uso de prototipos de tecnología básica, que definen el prototipo conceptual, ha mostrado resultados efectivos en:

- La migración desde la etapa de análisis a la etapa de diseño, al permitir entender y ubicar correctamente los asuntos de negocios de la empresa.
- La disminución en los tiempos de diseño, ya que permite enfocarse adecuadamente en la interfaz y su funcionalidad, y posponer los detalles de codificación hasta el momento oportuno.
- La posibilidad de contabilizar los objetos que integrarán una entidad web antes de la codificación, con lo cual es posible realizar estimaciones sobre los tiempos de desarrollo por interfaz.

La utilidad de los prototipos conceptuales se incrementa cuando se tiene un nuevo dominio del problema. Hacen posible la proyección de un tiempo inicial de desarrollo para el sistema o los subsistemas, al definir la complejidad de programación y poder asignar un número correcto de desarrolladores. También, es posible hacer estimaciones sobre los costos que implica el desarrollo del sistema, al conocer los requisitos de personal y los tiempos de desarrollo inicialmente obtenidos.

Pruebas de uso

Las pruebas de uso son un complemento esencial en el desarrollo de los prototipos conceptuales, ya que hacen posible su elaboración, revisión y aprobación por parte de los usuarios finales, permiten el entendimiento completo del dominio del problema y garantizan que los prototipos cumplan con la funcionalidad necesaria.

6.4.2. Prototipo detallado

El prototipo detallado se obtiene de manera directa del prototipo conceptual, sin embargo, entre mayor sea la disposición de recursos reutilizables, menor será el uso del prototipo conceptual, debido a que es posible utilizar la técnica del desarrollo rápido de aplicaciones.

RAD

El desarrollo rápido de aplicaciones debe ser manejado de forma correcta, ya que su principal característica, la codificación temprana, puede ayudar o perjudicar el desarrollo del sistema.

La forma correcta para aplicar el RAD es obtener la aprobación de la interfaz por parte del usuario, así como de las respectivas rutas de navegación, antes de iniciar cualquier actividad de programación. Los prototipos de tecnología básica permiten la incorporación de técnicas RAD sin ninguna dificultad y obtener el mejor rendimiento en la etapa de construcción del sistema.

6.5. Diseño de procesos intranet

6.5.1. Interacción de lógicas

El enfoque de interacción de lógicas permite eliminar la complejidad del diseño y construcción de un sistema, al tratar cada uno de los lenguajes de programación como un ámbito independiente que cumple con una función específica dentro de la entidad web o dentro de un proceso. Esto es posible al utilizar un lenguaje de definición de procesos para intranets (LDPI).

Lenguaje de definición de procesos intranet (LDPI)

El lenguaje de definición de procesos intranet (LDPI) permite realizar la especificación de las diversas lógicas, y es posible analizar los problemas inherentes (al eliminar los detalles de codificación).

LDPI al ser una especificación independiente del lenguaje empleado, permite incorporar nuevas tecnologías y lenguajes de programación, tal como applets de Java y tecnologías derivadas del SGML (como, XML y XSL).

6.6. Trabajos futuros

La Web tiene una dinámica muy alta, y las condiciones de desarrollo cambian rápidamente debido a nuevas tecnologías emergentes. Esto hace que los posibles trabajos a futuro y como continuidad y expansión del presente, sean enfocados a:

- La Web de objetos, es decir, el trabajo con objetos distribuidos y más específicamente con componentes, en donde se destaca un manejo elevado de scripts para su utilización.
- Refinamiento de elaboración de procesos intranet y construcción de árboles de procesos para identificar instancias del objeto en producción.
- Bases de datos orientadas a objetos, ya que cada una de las EW es considerada por la mayoría de las herramientas como un objeto web. De esta forma las EW podrían almacenarse en un OODBMS e interactuar con datos almacenados en un RDBMS.

Glosario

- **Applet** Una aplicación pequeña escrita en lenguaje Java que requiere un browser compatible con Java para ejecutarla.
- **Browser** Una aplicación que llama archivos HTML desde un servidor Web y visualiza el archivo formateado. También es llamado cliente.
- **CGI** Common Gateway Interface. Interfaz a los servidores HTTP la cual permite accesar las fuentes como son bases de datos o programas que residen fuera del servidor.
- **Cliente Web** Computadora que recibe la información de un servidor.
- **CORBA** Common Object Request Broker Architecture. Arquitectura para sistemas de objetos distribuidos definida por OMG.
- **CRC** Clases–Responsabilidades–Colaboradores.
- **CSS** Casacade Style Sheet.
- **Datamart** Bodega de datos que ha sido depurada la cual se utiliza para distribuir datos por temas en forma departamental y son sólo de consulta.
- **DHTML** Dinamic Hypertext Markup Language.
- **e–mail** Correo electrónico.
- **Evento** Una acción específica del usuario, como un clic del ratón, que es reconocida por el sistema.
- **EW** Entidad Web. Interfaz a la cual se hace referencia en un diagrama de hipervínculos.
- **frame** Una subventana dentro de una ventana del browser.
- **ftp** Un protocolo de transferencia de archivos empleado para la transferencia de archivos de texto o binarios de una computadora a otra sobre la red.
- **Gopher** Un protocolo de transferencia de archivos empleado para transferir archivos de texto o binarios de una computadora a otra sobre la red. El nombre original se debió a la pregunta del programa "go for".
- **Groupware** Grupos de trabajo en colaboración, centrados fundamentalmente en administración de documentos con multimedia, flujo de trabajo, correo electrónico, conferencias y planificación.

- **HTML** Hypertext Markup Language. Lenguaje de marcación de hipertexto empleado sobre la Web.
- **http** Protocolo de transferencia de hipertexto empleado para transferir archivos sobre Internet.
- **IDL** Interface Definition Language. Lenguaje para especificar interfaces de objetos independientes de las representaciones de un lenguaje de programación particular.
- **Internet** La red de redes que permite a personas en una computadora compartir información con personas en otra red ubicada en otra habitación o en el otro lado del mundo.
- **ITP** Internet Transaction Processing.
- **NSF** National Science Fundation.
- **OMA** Object Management Architecture. Es la arquitectura global y mapa de OMG, de la cual CORBA forma una parte.
- **OMG** Object Management Group. Consorcio de la industria internacional con más de 600 miembros que especifican un marco de trabajo orientado a objetos para computación distribuida, incluyendo CORBA.
- **ORB** Object Request Broker. El componente central de OMA el cual transmite solicitudes de invocación de operaciones a objetos distribuidos y devuelve los resultados al solicitante.
- **Perl** Lenguaje de programación que se emplea para escribir guiones CGI.
- **Protocolo** Descripción formal de formatos de mensajes y reglas que dos o más computadoras deben seguir para intercambiar mensajes.
- **PSI** Proveedores de Servicios de Internet.
- **Servidor Web** Un dispositivo que está dedicado a servir otros nodos conectados a la red. También, una aplicación que permite a archivos HTML enlazarse a través de la red.
- **SGML** Standard Generalized Markup Language.
- **SII** Sistema de Información Intranet.
- **UML** Unified Modeling Language.
- **World Wide Web** La parte de Internet que emplea el protocolo http para liberar información. La abreviatura es WWW o la Web.
- **XML** eXtensible Markup Language. Un subconjunto pequeño de SGML con una sintaxis simple para el marcado y declaración de la estructura de documentos.
- **XSL** EXtensible Style Laguage. Grupo de reglas para transformar un documento XML, a una estructura de representación.

Bibliografía

[Abl96] Ablan, J. ***"Developing Intranet Applications with JAVA"***. Sams Net. 1996.

[Bat94] Batini C., S. Ceri, S. B. Navathe. ***"Diseño Conceptual de Bases de Datos, Un Enfoque de Entidades–Interrelaciones"***. Addison–Wesley. 1994.

[Boo96] Booch G. ***"Análisis y Diseño Orientado a Objetos con Aplicaciones"***. 2ª edición. Addison–Wesley. 1996.

[Bro75] Brooks, F. ***"The Mythical Man–Month"***. Adison Wesley. 1975.

[Coa91] Coad, Peter and Edward Yourdon. ***"Object–Oriented Analisys"***. 2ª edition, Prentice–Hall. 1991.

[Coa91a] Coad, Peter and Edward Yourdon. ***"Object–Oriented Design"***. Prentice–Hall. 1991.

[Fow97] Fowler, M. with Kendall Scott. ***"UML Distilled, Applying the Standar Object Modeling Languaje"***. Addison–Wesley. 1997.

[Ger97] Gervae, N. and Peter Clark. ***"Developing Business Applications with OpenStep"***. Springer–Verlag. 1997.

[Hin97] Hinrichs R. J. ***"Intranets, What's the Bottom Line?"***. Sun Microsystems Press. Prentice–Hall. 1997.

[Jac99] Jacobson I., G. Booch, J. Rumbaugh. ***"The Unified Software Development Process"***. Addison Wesley. 1999.

[Kas96] Kassabgi G. ***"Using Progress V8"***. Que Corporation. 1996.

[Kor87] Korth H. F., A. Silberschatz. ***"Fundamentos de Bases de Datos"***. McGraw Hill. 1987.

[Law95] Lawton, S. ***"Intranets"***. Digital News & Review. April 1995.

[Orf96] Orfali, R., D. Harkey and J. Edwards. ***"The Essencial Client/Server Survival Guide"***. John Wiley & Sons, Inc. 1996.

[Pre93] [Pre98] Pressman, R. S. ***"Ingeniería del Software, Un Enfoque Práctico"***. 3ª. edición. McGraw Hill. 1993. y 4ª Edición. 1998.

[Rub97] Ruble, D. A. ***"Practical Analisys & Design for Client/Server & GUI Systems"***. Prentice–Hall. 1997.

[Tan97] Tanenbaum, A. S. ***"Redes de Computadoras"***. 3ª. Edición. Prentice–Hall.1997.

Referencias URL

[hBac] Bachiochi, D., et al. ***"Usability Studies and Designing Navigational Aids for the World Wide Web"***.

http://decweb.ethz.ch/WWW6/Technical/Paper180/Paper180.html

[hBow] Bowen, B. D. ***"4GLs discover the Web Web, Integrating Web with traditional database applications a hot topic for database tool makers"***.

http://developer.netscape.com/articles/swol–01–4gl.html

[hDMT] ***"Desktop Management Task Force"***.

http://www.dmtf.org/newsletter/newsletter_1.html

[hGel] Gellersen, H. W., Robert Wicke and Martin Gaedke. ***"WebComposition: An Object–Oriented Support System for the Web Engineering Lifecycle"***.

http://decweb.ethz.ch/WWW6/Technical/Paper232/Paper232.html

[hIng] Ingham, D., M. Little, S. Caughey and S. Shrivastava. ***"W3Objects: Bringing Object–Oriented Technology to the Web"***. Fourth International World Wide Web Conference. Boston, Massachusetts, 1995.

http://www.w3.org/pub/Conferences/WWW4/Papers2/141/

[hKes] Kesselring, M.. ***"XML As a Representation for Management Information– A White Paper"***.

http://www.dmtf.org/cim/cim_xml/XML_Whitepaper.htm

[hKey] Keyser, B. ***"DMTF backs XML standard"***.

http://www.infoworld.com/cgi–bin/displayStory.pl?981010.ehxml.htm

[hLau] St. Lauren, S. ***"A look at the adventages of XML"***.

http://cgi.chicago.tribune.com/1998/1998XML.html

[hLaz] Lazar, Z. P. and Peter Holfelder. ***"Web Database Connectivity with Scripting Languages Languages"***. Web Journal, Volume 2, Issue 2, Scripting Languages: Automating the Web.

http://developer.netscape.com/articles/index.html

[hTak] Takahashi, K. and Eugene Liang. ***"Analysis and Design of Web–based Information Systems"***. Sixth International World Wide Web Conference.

http://www6.nttlabs.com/HyperNews/get/PAPER245.html

[hVit] Vitale, F., Chao–Min Chiu and Michael Bieber. ***"Extending HTML In A Principled Way With Displets"***.

http://www6.nttlabs.com/HyperNews/get/PAPER155.html

Los abajo firmantes, integrantes de jurado para el examen de grado que sustentará el **Ing. Ulises Juárez Martínez,** declaramos que hemos revisado la tesis titulada:

"Metodología para el diseño de sistemas de información intranet"

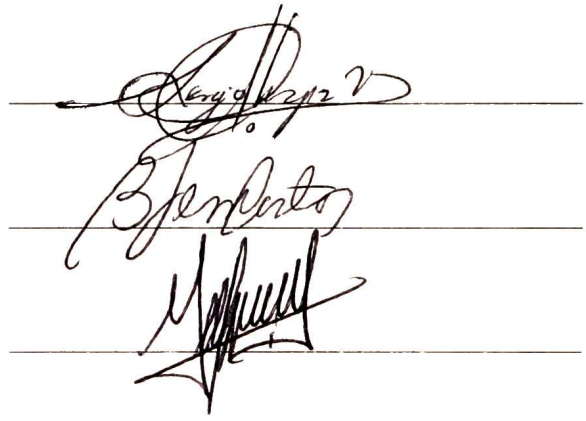
y consideramos que cumple con los requisitos para obtener el grado de Maestro en Ciencias, con especialidad en Ingeniería Eléctrica.

Atentamente

Dr. Sergio V. Chapa Vergara

Dr. Bárbaro J. Ferro

M. en C. Guillermo R. Domínguez de León



AUTOR JUAREZ MARTINEZ, U.

TITULO METODOLOGIA PARA EL DISEÑO DE SISTEMAS DE INFORMACION..

CLASIF. XM 99.13 **RGTRO.** BI 15633

NOMBRE DEL LECTOR	FECHA PREST	FECHA DEVOL
FB 00000139770		
Arturo Zuñiga L.	3/05/2000	25/5/00
Carlos Campos B.	29/V/2000	1-X-00
Giner Alor Hdez	26/junio/2001	2/VII/01
Salvador Castro Enciso	24/oct/02	13/11/02
Carlos Silva Sánchez	13/Nov/02	2-XII-02
Eli		