UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE UN SITIO WEB BASADO EN PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE Y MOPROSOFT

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N :

FLORES MORALES RAÚL

RAMÍREZ HERNÁNDEZ JOSÉ RENÉ

DIRECTOR
DRA. ANGÉLICA DEL ROCÍO LOZANO CUEVAS

CIUDAD UNIVERSITARIA MÉXICO, D. F., 2006

Agradecimientos

He llegado a la conclusión de un paso importante y trascendente en mi vida, sin embargo no sería posible sin la unión de esfuerzos y por eso le agradezco a mis papás Lilia y René, todo el apoyo, cariño y cobijo, que como su hijo me han dado, no solo en esta última etapa universitaria, sino desde mis primeros pasos en la escuela, sin su unión nada sería lo mismo y el final de esta etapa no sería tan feliz como éste. A mis hermanos Toño e Israel, que me han acompañado como verdaderos amigos, su ayuda, comprensión y tolerancia las he apreciado como no tienen idea, échenle ganas. No se nos olvide que es un triunfo de todos, sigamos adelante juntos por aquellos que están por venir.

Quiero mencionar también un agradecimiento profundo a mis familiares por sus buenos deseos y apoyo moral, a mis profesores por su labor y compromiso, a la Facultad de Ingeniería y a la UNAM por darme una educación profesional, al LTST y a la Dra. Angélica Lozano por darme la oportunidad de colaborar este tiempo con ellos.

No quiero dejar pasar la oportunidad de compartir esta alegría con mis amigos y amigas de la Facultad, del LTST y amistades muy especiales con las que he contado de su compañía durante mucho tiempo. Sin embargo un agradecimiento especial a Raúl, por haber trabajado duro para sacar este proyecto adelante y también a Alex por el gran respaldo que nos brindó para la realización de la tesis.

Sin embargo, el crédito más importante es de alguien de quien siempre me apoyé y le doy gracias a él, por iluminarme, guiarme y levantarme en los momentos de adversidad, Dios te pido me sigas acompañando.

José René Ramírez Hernández

Agradecimientos

Este es un paso mas en mi vida, un momento en el que culmino mis estudios, durante toda mi vida he recibido el apoyo de dos personas sin las cuales no sería la persona que soy hoy mis padres; Raúl Flores Ibarra y María Eugenia Morales Flores, gracias por ser los más grandes pilares de mi vida, por darme todo su apoyo en todos los momentos en los que más los necesitaba, gracias por estar siempre a mi lado en cada decisión que me llevo finalizar satisfactoriamente mi carrera, por todo el amor y comprensión que siempre recibí de ustedes. Gracias Papas por depositar toda su confianza y cultivarme como lo han hecho; sin ustedes nada de esto hubiera sido posible.

A mis tres mujeres, mis hermanas Leticia, Rosa Isela y Anabell quienes han compartido conmigo todos los esfuerzos, angustias y éxitos, gracias por existir.

A Karmina Estrada mi niña hermosa gracias por animarme y estar conmigo en este momento tan importante de mi vida, por regalarme lo mas bonito de la vida, Tu Amor. Gracias a tu familia por todo su apoyo.

Mi agradecimiento a la Facultad de Ingeniería y a la UNAM por darme una herramienta muy importante en mi vida, mi formación académica. Gracias al Laboratorio de Transporte y Sistemas Territoriales del Instituto de Ingeniería de la UNAM por todo el apoyo para la realización de este trabajo, muy en especial a nuestra directora de tesis la Dra. Angélica Lozano y a Alejandro Guzmán, Gracias Alex por todo tu apoyo y dedicación.

Gracias a mis amigos por haber estado conmigo durante toda la carrera muy en especial gracias a René por todo su apoyo, empeño y amistad que fueron de gran importancia para la elaboración de este trabajo.

Estoy seguro de que tú también has estado conmigo, como lo has hecho durante toda mi vida, Gracias Dios por permitirme concluir una etapa más de mi vida, gracias por darme la virtud de estar rodeado de buenas personas.

Raúl Flores Morales

Introducción

1. LENGUAJE UNIFICADO DE MODELADO

Antecedentes

Componentes de UML
- Bloques de Construcción
  - Elementos
  - Relaciones
  - Diagramas
- Reglas de UML
- Mecanismos comunes de UML

2. PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE

Historia del Proceso Unificado
- El método Ericsson
- Proceso Objectory
- El Proceso Objectory de Rational
- El Proceso Unificado de Rational.

El Proceso Unificado
- Dirigido por casos de Uso
- Centrado en la arquitectura
- Iterativo e Incremental

La Vida del Proceso Unificado

FLUJOS DE TRABAJO FUNDAMENALES DEL PROCESO UNIFICADO
- Flujo de Trabajo de Requerimientos
- Flujo de Trabajo de Análisis
- Flujo de Trabajo de Diseño
- Flujo de Trabajo de Implementación
- Flujo de Trabajo de Prueba

Fases del Proceso Unificado
- Fase de Inicio
- Fase de Elaboración
- Fase de Construcción
- Fase de Transición

3. MOPROSOFT

Procesos de Software

Antecedentes de MoProSoft
- La estrategia 6 de ProSoft
- Componentes de la norma mexicana

Estructura de MoProSoft
- Elementos globales
- Los Procesos de MoProSoft

4. APLICACIONES DINÁMICAS DE INTERNET

Definición

Construyendo una RIA
- La tecnología por parte del cliente
- La tecnología por parte del servidor

Introducción

Cuando se debe realizar un proyecto relacionado con el desarrollo o mantenimiento de software, las personas que tienen la responsabilidad de llevar a cabo el proyecto, deben de contestar algunas de las siguientes preguntas: cómo se va a hacer, con qué se va a hacer, cuándo se va a hacer, qué se necesita para desarrollarlo, etc.

Varios estudios y análisis respecto al éxito alcanzado en los proyectos de software, se ha detectado que cerca de un 70% de ellos no logra los objetivos trazados. Con el paso del tiempo, personas involucradas en la computación se han preocupado por crear modelos, lenguajes, procesos, metodologías, herramientas y formas de gestión que ayudan a desarrollar los proyectos de software. A la disciplina que ha reunido estos conocimientos se le dio el nombre de Ingeniería del Software, ésta, fomenta un trabajo ordenado, controlado, estandarizado y un trabajo en equipo, toma en cuenta las condiciones de las organizaciones y el potencial creativo de las personas y con el objetivo primordial de obtener productos de software de calidad.

En el mundo, desde hace varios años los desarrolladores de software usaron a la Ingeniería de Software como una forma de trabajo, pero principalmente la adoptaron como una disciplina, puesto que al ponerla en práctica, los desarrolladores, ahora llamados ingenieros de software, utilizan procesos, métodos y técnicas probadas.

En un mundo donde la competencia se da día a día, se han creado estándares internacionales de calidad y marcos de referencia de trabajo como el Modelo de Madurez de Capacidades o ISO 9001-2000, los cuales son un marco de referencia de trabajo para las organizaciones, al mismo tiempo se han creado las certificaciones para éstos, las cuales avalan que una empresa sigue ese marco de trabajo y tiene un cierto grado de madurez, lograr una certificación no sólo cuesta adoptar una cultura laboral, también cuesta dinero.

En el ámbito de la competitividad mundial, las organizaciones mexicanas, llámense instituciones, empresas grandes o pequeñas, se han enfrentado a dos problemas: el primero consiste en que la Ingeniería de Software comienza a ser usada, no por desconocimiento de ésta, sino porque apenas se está adoptando como una forma de trabajo; la segunda es que es sumamente difícil, obtener una certificación por lo costosa que resulta, más tratándose de que en el país, las empresas desarrolladoras de software son pequeñas.

El gobierno de México, por medio de la Secretaría de Economía ha creado un marco de de referencia llamado MoProSoft, el cual es equiparable con los modelos o marcos ya mencionados y que se rigen a nivel mundial como resultado de estudios que le permitieron conocer que el nivel de madurez de las empresas en el país tenían un nivel de 0.9 de una escala internacional de de 0 a 5. MoProSoft pretende ser norma y el cual al ser basado en el estándar ISO 15504, estándar del cual también se basan otros modelos y que por lo mismo, manejan prácticas, actividades y procesos equiparables, lo convierte en una vía alternativa para que todas aquellas empresas que lo introduzcan en sus procesos de desarrollo de software, alcancen un nivel estandarizado de calidad en sus productos y que potencialmente, una vez que sea norma mexicana, sea un potencial estándar para Latinoamérica y permita una competencia a nivel mundial.

El motivo de este trabajo de tesis, llamado Desarrollo de un Sitio Web basado en Proceso Unificado de Desarrollo de Software y MoProsoft, permite demostrar que todo aquello que engloba la Ingeniería de Software, adaptado a un contexto y a una situación particular es posible. En el desarrollo de software, se pueden hablar de tres grandes rubros:

- Un marco de referencia para la calidad y madurez del proceso utilizado en el desarrollo del software.

- Un proceso que guíe en la construcción de software.

- Un lenguaje de modelado que sirva para comunicar diversos aspectos del software que se esté desarrollando.

El objetivo principal es desarrollar un sitio en Internet para el Laboratorio de Transporte y Sistemas Territoriales del Instituto de Ingeniería de la UNAM. Se adoptará el marco de referencia MoProSoft, puesto que se utilizarán prácticas usadas internacionalmente. Se usará Proceso Unificado como metodología para el desarrollo de software, puesto que es adaptable, para el tipo de proyecto, en tiempo y en actividades. Finalmente se usará UML como el lenguaje de modelado, pues es el más usado por la comunidad informática en el mundo.

Este trabajo se ha seccionado en tres partes: marco teórico, desarrollo y resultados. Dentro del marco teórico se pondrán en contexto los conceptos que se emplearán durante la creación del producto, como uso de estereotipos en UML, las fases por las que atraviesa el software que involucra el proceso unificado y los documentos generados y roles desempeñados en MoProSoft, además de las tecnologías empleadas como J2EE y Flash. La parte de desarrollo es el cómo se fue creando el sitio Web, desde las etapas de las reuniones con los clientes, pasando por el diseño, el uso de las tecnologías, las pruebas, las mejoras, etc. Por último, se tiene la parte de los resultados, en donde se hará un análisis de los resultados logrados y se mostrará la documentación que avale que se han seguido las prácticas pedidas por MoProSoft y que avalen un producto de calidad.

Se espera alcanzar un resultado satisfactorio para ambas partes, clientes y desarrolladores, pero también se pretende demostrar que los desarrolladores de software, sean un equipo de trabajo pequeño o grande, pueden desarrollar bajo estándares de calidad internacionales, sin importar el contexto, académico o profesional, y que puede adoptarse una cultura y disciplina de trabajo que eleva el porcentaje de éxito.

1. LENGUAJE UNIFICADO DE MODELADO

En un proyecto de software suele emplearse el llamado triángulo del éxito en donde se explican los componentes necesarios para el desarrollo: notación (referente a la documentación), proceso (referente al "cómo" se hacen las cosas) y herramientas (es el "con qué" se realizan). Se deben usar los tres componentes en conjunto para obtener soluciones al software que se quiere desarrollar. El Lenguaje Unificado de Modelado UML (Unified Modeling Lenguaje), por sus siglas en inglés, es capaz de sintetizar este funcionamiento conjunto.

UML es un lenguaje gráfico que permite visualizar, especificar, construir, documentar, diseñar, estructurar y realizar una abstracción del sistema y sus componentes.

UML no debe confundirse con una metodología para modelar, este lenguaje es una herramienta que permite, en combinación con una metodología formal, realizar el modelo de un software, particularmente aquellas metodologías relacionadas con la orientación a objetos. UML al ser un lenguaje, cuenta con una sintaxis y una semántica.

El presente capítulo desarrollará el aspecto teórico que comprende a UML. Se inicia con una breve mención de los antecedentes históricos, objetivos y funcionalidad que persigue este lenguaje; terminando los antecedentes, se iniciará con el desarrollo de los componentes de UML y que están relacionados a la parte técnica del modelado, estos componentes son los *Bloques de Construcción*, donde se desarrolla la estructura de UML y los elementos que la integran, las *Reglas de UML*, para lograr la consistencia del modelado y los *Mecanismos Comunes*, los cuales permiten extender el lenguaje de UML. Conforme se desarrolle el capítulo se irá detallando cada uno de ellos. Al finalizar el capítulo, se pretende concentrar los puntos más importantes de UML con el objetivo de aplicarlos de la mejor manera posible en el capítulo correspondiente al desarrollo del sitio Web.

1.1. Antecedentes

Como parte de su historia, UML fue creado por Grady Booch, James Rumbaugh e Ivar Jacobson en Rational Corporation® (Recientemente adquirido por IBM), los cuales reunieron y unificaron las tres metodologías de diseño y análisis orientado a objetos más reconocidos:

- Metodología de Grady Booch para la descripción de conjuntos de objetos y sus relaciones.
- Técnica de modelado orientada a objetos de James Rumbaugh (OMT: Object-Modeling Technique).
- Aproximación de Ivar Jacobson (OOSE: Object- Oriented Software Engineering) mediante la metodología de casos de uso (use case).

UML tiene como objetivos, entre otros:

- Modelar sistemas, desde el concepto hasta su ejecución.

- Ser válido para cualquier sistema.
- Ser utilizable tanto por personas como por máquinas.
- Incluir conceptos necesarios para utilizar procesos iterativos, para resolver requisitos dirigidos por casos de uso.
- Ser suficiente para manejar conceptos de concurrencia, distribución, encapsulación y manejo de componentes.
- Ser un lenguaje universal.
- Imponer un estándar mundial.

Así, UML fue creado como un estándar de modelado, el cual es aceptado por la comunidad informática. En UML se fomenta un proceso guiado por casos de uso, centrado en una arquitectura, y es iterativo e incremental.

El modelado mediante UML se hace con el fin de que sea comprensible y entendible, por aquellos que estén involucrados en el desarrollo del software, incluyendo a clientes, usuarios y desarrolladores. Un modelo es una simplificación de la realidad creada para comprender mejor un sistema, una abstracción del modelo es obtener la parte esencial del sistema con un cierto grado de detalle. UML utiliza modelos orientados a objetos, lo cual quiere decir que representa la realidad a partir de objetos, con características propias o heredadas, y que éstos interactúan entre sí por lo que está constituido. Un modelo orientado a objetos sirve, entre otras cosas para:

- visualización de cómo es el sistema, su estructura y componentes;
- mantenimiento de sistemas;
- reducción de riesgos;
- documentación;
- representación fiel de la realidad; y
- ser semánticamente consistente.

1.2. Componentes de UML

UML está formado por tres componentes principales: Bloques de construcción, Reglas y Mecanismos Comunes, en ellos se encierra el marco conceptual de UML. El diagrama de la Figura 1.1 muestra la estructura de UML.

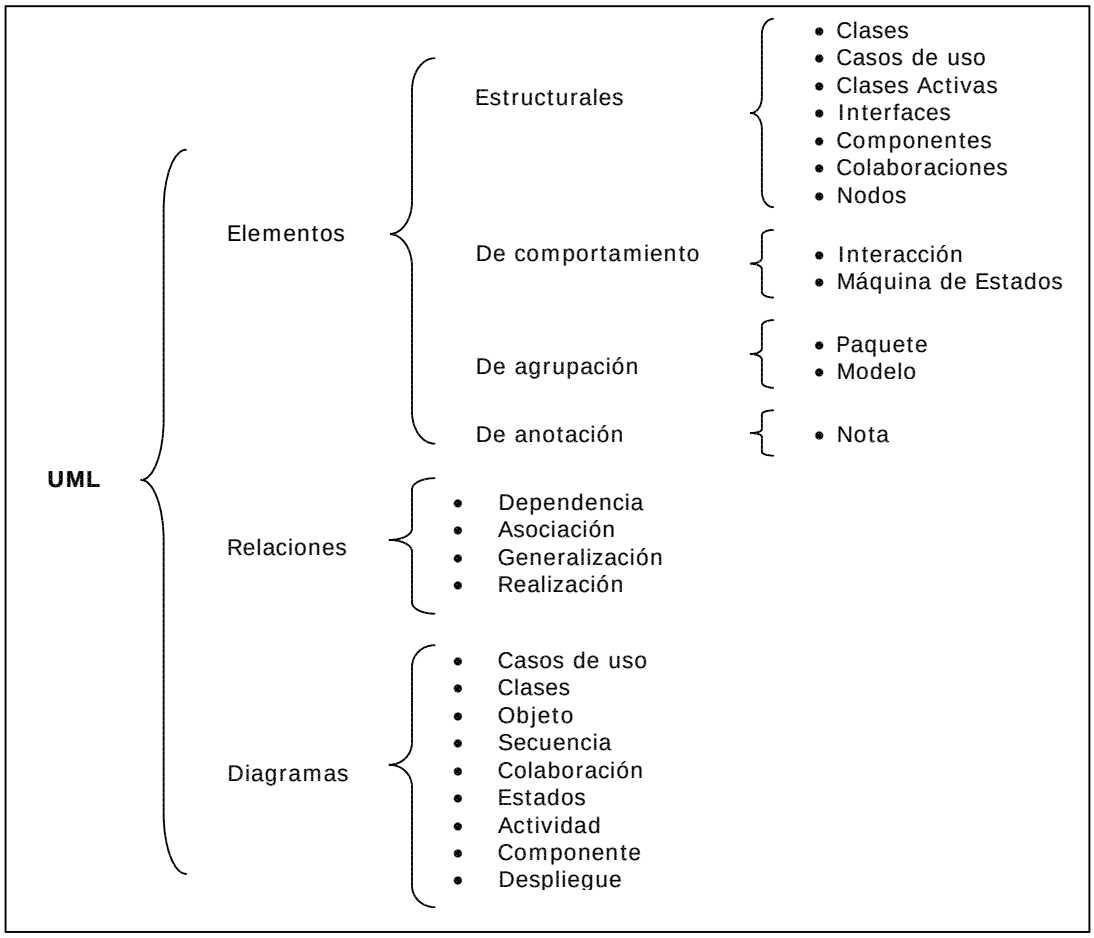


Figura 1.1 Estructura de UML.
Fuente: Ivar Jacobson, "El Lenguaje Unificado de Modelado, Manual de Referencia", 1999

1.2.1. Bloques de Construcción

Los Bloques de Construcción son de tres tipos: *elementos* (unidades básicas y abstractas para el modelado, se subdividen en estructurales, de comportamiento, de agrupación y de anotación), *relaciones* (actúan como la unión entre los distintos elementos) y *diagramas* (conjunto de elementos y relaciones que representan el sistema modelado).

1.2.1.1. Elementos

Elementos Estructurales

Los Elementos Estructurales se refieren a las partes estáticas de un modelo, que representan conceptos o materiales. Éstos se subdividen en: Caso de Uso, Clases, Clases Activas, Interfaz, Componentes, Colaboraciones y Nodo. A continuación se describen cada uno de ellos.

Caso de Uso

Son las descripciones de las operaciones o tareas específicas que se realizan tras una petición de un actor u otro Caso de Uso, produciendo un resultado observable, dejando en claro que no define cómo lo hace. Los Casos de Uso se emplean para estructurar aspectos de comportamiento de un modelo y su representación se muestra en la Figura 1.2. En un Caso de Uso, pueden aparecer descripciones o secciones, las cuales generalmente son:

- Descripción (lo que hace el caso de uso).
- Precondiciones (condiciones que se deben cumplir antes de poder utilizar el caso de uso).
- Postcondiciones (condiciones que se deben cumplir después de ejecutar las acciones del caso de uso).
- Flujo de eventos (describe, paso a paso, lo que sucede en el caso de uso).

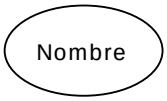


Figura 1.2 Representación de Caso de Uso.

Un elemento importante que vale la pena mencionar, por ser fundamental en el manejo de los Casos de Uso, es el actor. Un Actor es un rol que juega una persona, hardware o software hacia el sistema. El actor no necesariamente representa a una persona en particular, sino más bien es la labor que realiza entorno al sistema. Hay tres tipos principales de actores: usuarios del sistema, otros sistemas que interactúan con el sistema que está siendo modelado y el tiempo (Figura 1.3).

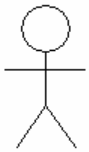


Figura 1.3 Representación de Actor.

Clase

Son un conjunto de objetos que comparten características comunes, como atributos, semántica, operaciones y relaciones (Figura 1.4).

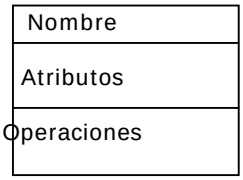


Figura 1.4 Representación de una Clase.

Clases Activas

Contienen uno o más procesos, los que pueden originar nuevas actividades debido a que tienen uno o más procesos de ejecución. Una Clase Activa es igual que una clase, excepto que sus objetos representan elementos cuyo comportamiento es concurrente con otros elementos[1] (
Figura 1.5).

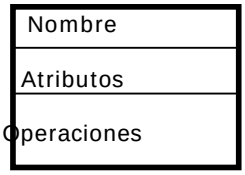


Figura 1.5 Representación de una Clase Activa.
El contorno de las líneas es más grueso que el de una clase "normal".

Como un elemento importante de las Clases Activas, se tiene al Objeto. Un Objeto es una instancia de una clase, es una manifestación concreta de una abstracción, que encapsula un estado y un comportamiento (Figura 1.6).

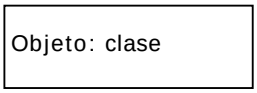


Figura 1.6 Representación de un Objeto.

Interfaz

Es la descripción del servicio que ofrece una clase o componente, definiendo un conjunto de especificaciones o métodos del servicio, pero no así, la forma en que será implementado. Una interfaz representa el comportamiento completo o una parte, gráficamente, suele estar acompañada con una conexión a su clase o componente que la invoca. Sus dos formas de representación son como las mostradas en la Figura 1.7 y Figura 1.8

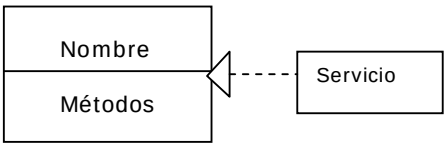


Figura 1.7 Representación en su forma canónica.

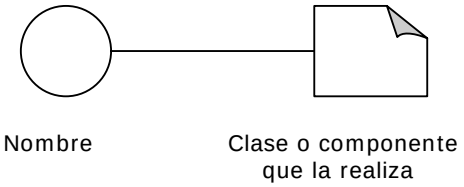


Figura 1.8 Representación en su forma abreviada.

Componentes

Se refiere al empaquetamiento físico de clases, interfaces y colaboraciones que implementan la funcionalidad del sistema y también incluye a las instancias de objetos encargadas del estado de éste. Son una parte física y reemplazable (Figura 1.9).

1 Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Lenguaje Unificado de Modelado"; Adisson Wesley Iberoamericana, Madrid 1999.

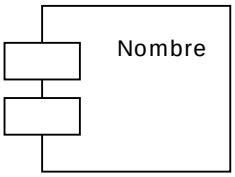


Figura 1.9 Representación de un Componente.

Colaboraciones

Son interacciones entre elementos que funcionan con base en una cooperación y cuyo resultado muestra aspectos estructurales y de comportamiento del sistema (Figura 1.10).

Figura 1.10 Representación de una Colaboración.

Nodo

Es un elemento físico existente en tiempo de ejecución y representa un recurso computacional, generalmente con capacidad de procesamiento y memoria (Figura 1.11).

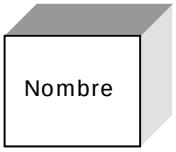


Figura 1.11 Representación de un Nodo.

Elementos de Comportamiento

Los Elementos de Comportamiento son la parte dinámica de UML, la cual representa el comportamiento en tiempo y espacio. Hay dos tipos: Interacciones y Máquina de Estados. A continuación se detalla cada uno de ellos.

Interacciones

Es un mensaje que va de un objeto a otro, el cual tienen un fin específico y un contexto particular. Una Interacción involucra muchos otros elementos, incluyendo mensajes, secuencias de acción (el comportamiento invocado por un mensaje) y enlaces (conexiones entre objetos)[2] (Figura 1.12).

2 Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Lenguaje Unificado de Modelado"; Adisson Wesley Iberoamericana, Madrid 1999

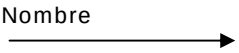


Figura 1.12 Representación de una Interacción.

Máquina de Estados

Especifica una secuencia de estados por los que atraviesa un objeto o una interacción, así como su respuesta a estos estados. Una Máquina de Estados se une a una clase y describe, generalmente, la respuesta de una instancia de la clase, a los eventos que recibe. Las máquinas de estados también se pueden unir a las operaciones, a los casos de uso, y a las colaboraciones para describir su ejecución[3]. Un *evento*, dentro del contexto de la máquina de estados es la aparición de una respuesta que dispara una transición de estado (Figura 1.13).

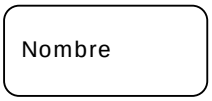


Figura 1.13 Representación de un Estado.

Elementos de Agrupación

Los Elementos de Agrupación son los elementos que proporciona UML para organizar el modelo, por ejemplo para fraccionar el modelo. Lo componen los elementos Paquete y Modelo. A continuación se detallan ambos.

Paquete

Organiza elementos en grupo, incluyendo elementos estructurales, elementos de comportamiento y componentes que sólo existen en tiempo de ejecución (Figura 1.14).

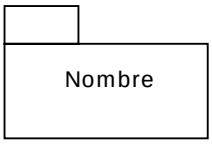


Figura 1.14 Representación de un Paquete.

Modelo

Un modelo es un paquete que abarca una descripción completa de una vista particular de un sistema. Proporciona una descripción cerrada de un sistema a partir de un punto de vista. No tiene dependencias fuertes en otros paquetes, tales como dependencias de implementación o dependencias de herencia. Generalmente se estructura en forma de árbol[4].

Elementos de Anotación

Los Elementos de Anotación sirven para explicar, describir, hacer observaciones, etc. sobre cualquier elemento que conforma el modelo. El elemento Nota es el único que pertenece a

3 Rumbaugh, J.; Jacobson, I.; Booch, G.; "El Lenguaje Unificado de Modelado, Manual de Referencia"; Pearson Educacion S.A., Madrid 2000.
4 Rumbaugh, J.; Jacobson, I.; Booch, G.; "El Lenguaje Unificado de Modelado, Manual de Referencia"; Pearson Educacion S.A., Madrid 2000.

esta categoría, su función es mostrar textualmente comentarios y restricciones junto a elementos o un grupo de éstos (Figura 1.15).

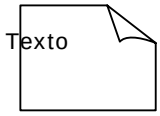


Figura 1.15 Representación de una Nota.

1.2.1.2. Relaciones

Una relación es la unión entre los distintos elementos. Son de cuatro tipos: Dependencia, Asociación, Generalización y Realización. A continuación se detalla cada uno de ellos.

Dependencia

Es una relación entre dos elementos, uno dependiente y otro independiente, en donde un cambio puede afectar a otro. La línea va del elemento dependiente al independiente (Figura 1.16).

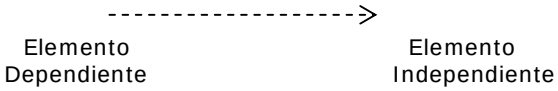


Figura 1.16 Representación de una Relación de Dependencia.

Asociación

Es una relación estructural que resume un conjunto de enlaces entre objetos. Incluyendo por ejemplo, que objetos pertenecientes a una misma clase, estén relacionados entre sí. El símbolo puede llevar nombre y algún identificador de los elementos que asocia (Figura 1.17).

Figura 1.17 Representación de una Relación de Asociación.

Generalización

Hace referencia a la relación donde los objetos de un elemento general, pueden ser sustituidos por los objetos de un elemento especializado, pues estos elementos tienen la misma estructura y comportamiento del elemento general (Figura 1.18).

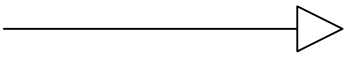


Figura 1.18 Representación de una Relación de Generalización.

Realización

Es una relación en donde se garantiza que una parte realizante cumple con una serie de especificaciones dada por un clasificador (Figura 1.19).

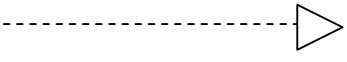


Figura 1.19 Representación de una Relación de Realización.

1.2.1.3. Diagramas

Es la agrupación lógica del conjunto de elementos, los cuales representan el modelo desde diferentes vistas. Todos ellos se agrupan bajo un área, un tipo de vista, una función, un manejo de objetos y una forma de empleo.

Área Conceptual. Puede agruparse en estructura estática, estructura dinámica y gestión de modelo. La estructura estática define los conceptos claves de la aplicación, sus propiedades internas y las relaciones entre cada una de éstas. La estructura dinámica involucra aspectos relacionados al comportamiento (detallar mas). Las estructuras, estática y dinámica, están compuestas mediante vistas. Una vista es un subconjunto de UML que modela construcciones que representan un aspecto de un sistema. Las vistas se clasifican en[5]:

- Vista estática. Modela los conceptos del dominio de la aplicación. No describe el comportamiento del sistema dependiente del tiempo.
- Vista de casos de uso. Modela la funcionalidad del sistema según los actores y su participación en los casos de uso.
- Vista de interacción. Describe secuencias de intercambios de mensajes entre los diferentes roles que definen el comportamiento del sistema. La vista se complementa con dos diagramas, el diagrama de secuencia y el de colaboración.
- Vista de máquina de estados. Modela las historias de vida de un objeto, haciendo uso de estados y transiciones para ejecutar ciertas acciones.
- Vista de actividades. Describen grupos de secuencias de actividades, describe el comportamiento de alto nivel de la ejecución de un sistema.
- Vista de implementación. Modela los componentes de un sistema, así como las dependencias entre los componentes.
- Vista de despliegue. Representa la disposición de las instancias de los componentes de ejecución en instancias de nodos.
- Vista de gestión de modelo. Modela la organización del modelo den sí mismo.

Los diferentes tipos de Vistas se auxilian de diagramas de UML para ser representadas, estos diagramas están clasificados dentro de las siguientes categorías: *Diagramas de comportamiento, Diagramas de interacción, Diagramas estructurales y Diagramas de implementación*. A continuación se mencionan los diagramas específicos que integran cada una de éstas.

Diagramas de comportamiento (modelan cómo se comporta el sistema). Esta clasificación está compuesta por los siguientes diagramas:

Diagramas de Casos de Uso

- **Área**: Estructural.
- **Vista:** Vista de Casos de Uso.
- **Función:** Muestra actores y formas en que éstos pueden utilizar el sistema.
- **Conceptos asociados:** Caso de uso, actor, asociación, extensión y generalización.

5 Rumbaugh, J.; Jacobson, I.; Booch, G.; "El Lenguaje Unificado de Modelado, Manual de Referencia"; Pearson Educacion S.A., Madrid 2000

- **Empleo:** Modelar el contexto de un sistema, modelar los requisitos del sistema.

Diagramas de Estados

- **Área:** Dinámica.
- **Vista:** Vista de Estados de máquina.
- **Función:** Muestra los diferentes estados por los que puede pasar un objeto, así como las transiciones y eventos asociadas.
- **Conceptos asociados:** Estado, evento, transición y acción.
- **Empleo:** Modelar objetos reactivos.

Diagramas de Actividad

- **Área:** Dinámica.
- **Vista:** Vista de actividad.
- **Función:** Muestra el flujo de control entre objetos.
- **Conceptos asociados:** Estado, actividad, transición, determinación, división y unión.
- **Empleo:** Modelar un flujo de trabajo, modelar una operación.

Diagramas de interacción (muestran cómo interaccionan entre sí objetos de un sistema). Esta clasificación está compuesta por los siguientes diagramas:

Diagramas de Secuencia

- **Área:** Dinámica.
- **Vista:** Vista de interacción.
- **Función:** Muestra cómo se mandan mensajes los actores y objetos de un sistema a lo largo del tiempo.
- **Conceptos asociados:** Interacción, objeto, mensaje y activación.
- **Empleo:** Modelar flujos de control por ordenación temporal.

Diagramas de Colaboración

- **Área:** Dinámica.
- **Vista:** Vista de interacción.
- **Función:** Muestra las relaciones existentes entre actores y objetos, y los mensajes enviados entre ellos.
- **Conceptos asociados:** Colaboración, interacción, rol de colaboración y mensaje.
- **Empleo:** Modelar flujos de control por organización.

Diagramas estructurales (modelan algún aspecto de la estructura del sistema). Esta clasificación está compuesta por los siguientes diagramas:

Diagrama de Clases

- **Área:** Estructural.
- **Vista:** Vista Estática.
- **Función:** Muestra la estructura de clases de un sistema, incluyendo las relaciones que pudieran existir entre ellas.
- **Conceptos asociados:** Clase, asociación, generalización, dependencia, realización e interfaz.
- **Empleo:** Modelar el vocabulario de un sistema, modelar colaboraciones simples, modelar un esquema lógico de base de datos.

Diagramas de Objeto

- **Área:** Estructural.
- **Vista:** Vista de interacción.
- **Función:** Muestra un conjunto de objetos y sus relaciones (enlaces) en un instante determinado. Equivale a una instancia de un diagrama de clases (o una parte del mismo) y se utiliza generalmente para documentar estructuras de datos complejas.
- **Conceptos asociados:** Objetos o instancias.
- **Empleo:** Modelar estructuras de objetos.

Diagramas de implementación (muestran aspectos relacionados con la implementación). Esta clasificación está compuesta por los siguientes diagramas:

Diagramas de Componentes

- **Área:** Estructural.

- **Vista:** Vista de Implementación.
- **Función:** Muestra cómo se organizan los elementos constituyentes del sistema y las dependencias existentes entre ellos.
- **Conceptos asociados:** Componente, interfaz, dependencia y realización.
- **Empleo:** Modelar código fuente, modelar versiones ejecutables, modelar bases de datos físicas, y modelar sistemas adaptables.

Diagramas de Despliegue

- **Área: Estructural.**
- **Vista:** Vista de Despliegue**.**
- **Función:** Muestra la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.
- **Conceptos asociados:** Nodo, componente, dependencia, localización.
- **Empleo:** Modelar código fuente, modelar versiones ejecutables, modelar bases de datos físicas, y modelar sistemas adaptables.

1.2.2. Reglas de UML

Las reglas de UML son una serie de reglas para lograr que los modelos sean semánticamente autoconsistentes y armónicos con otros modelos relacionados. Las reglas semánticas se aplican a:

- Los nombres: cómo llamar a los elementos, relaciones y diagramas.
- Alcance: contexto que da un significado específico a un nombre.
- Visibilidad: cómo se pueden ver y utilizar estos nombres por otros.
- Integridad: cómo se relacionan apropiadamente y consistentemente unos elementos con otros.
- Ejecución. qué significa ejecutar o simular un modelo dinámico.

1.2.3. Mecanismos comunes de UML

Los mecanismos de UML son patrones de características comunes que se aplican de forma consistente a través de todo el lenguaje; si no existe algún elemento de UML para representar alguna característica del sistema, UML permite extender el lenguaje de forma controlada. Los estereotipos, los valores etiquetados y las restricciones son los mecanismos que proporciona UML para añadir bloques de construcción, crear nuevas propiedades y especificar una nueva semántica[6]. UML utiliza cuatro mecanismos: Especificaciones, Adornos, Divisiones Comunes y Mecanismos de extensibilidad.

6 Rumbaugh, J.; Jacobson, I.; Booch, G.; "El Lenguaje Unificado de Modelado, Manual de Referencia"; Pearson Educacion S.A., Madrid 2000.

- Especificaciones. Es una explicación textual de la sintaxis y la semántica de un bloque de construcción. La notación gráfica de UML se utiliza para visualizar un sistema, y la especificación de UML se utiliza para enunciar los detalles del sistema.

- Adornos. Se refiere a adornos gráficos o textuales que especifican información detallada de los elementos de UML.

- Divisiones Comunes. Cuando se modelan sistemas orientados a objetos, es posible dividirlos de dos formas: a) División entre Clase y Objeto, gráficamente, UML distingue un objeto utilizando el mismo símbolo de la Clase y subrayando el nombre; y b) División entre Interfaz e Implementación, una Interfaz declara un contrato, y una Implementación representa una realización concreta de ese contrato del objeto.

- Mecanismos de Extensibilidad. UML permite extender el lenguaje de manera controlada; los Mecanismos de Extensibilidad se utilizan para adaptar UML a las necesidades específicas del software. Los mecanismos de extensión incluyen: Estereotipos, Valores etiquetados, Restricciones.

    - Estereotipos. Extiende el vocabulario de UML permitiendo crear nuevos tipos de bloques de construcción que deriven de los existentes y que sean específicos a un problema. Un estereotipo se representa como un nombre entre comillas tipográficas (por ejemplo <<nombre>>) y se coloca sobre el nombre de otro elemento.

    - Valores etiquetados. Extiende las propiedades de un bloque de construcción de UML, permitiendo añadir nueva información a la especificación del elemento. Un valor etiquetado se ve como una cadena de caracteres entre llaves que se coloca bajo el nombre de otro elemento. Esta cadena incluye un nombre (etiqueta), un separador (el símbolo "=") y un valor (de la etiqueta).

    - Restricciones. Extiende la semántica de un bloque de construcción de UML, permitiendo añadir nuevas reglas o modificar las existentes. Una restricción especifica las condiciones que deben cumplirse para que el modelo esté bien formado.

El estándar UML es muy amplio, por lo que aquí no se presentan todas sus particularidades. Se recomienda consultar la bibliografía si se desea completar los conocimientos o profundizar en algún aspecto concreto.

Como se mencionó en la introducción del capítulo, se han desarrollado concretamente los componentes fundamentales de UML: Bloques de Construcción, Reglas y Mecanismos de extensibilidad. En el capítulo Desarrollo del Sitio Web LTST, se harán uso de muchos de los elementos que se manejan en este capítulo, como diagramas, los elementos que lo componen, las reglas de UML y también se hará uso de los mecanismos de extensibilidad, debido a aún no está generalizado y estandarizado el desarrollo de la tecnología Flash (la cual será desarrollada en el capítulo Aplicaciones Dinámicas de Internet, dentro de este Marco Teórico) con UML.

El siguiente capítulo consistirá en el desarrollo del Proceso Unificado de Desarrollo de Software, el cual es una guía para implementar el uso de UML y usarlo como herramienta en el modelado de los diferentes sistemas de software por medio de una serie de tareas y prácticas enmarcadas en flujos de trabajo y etapas.

Este capítulo expone una de las bases teóricas de este proyecto: El Proceso Unificado de Desarrollo de Software. Un proceso es un conjunto de actividades técnicas y administrativas realizadas durante la adquisición, desarrollo, mantenimiento y retiro de software.[7] Una metodología de ingeniería del software es un proceso para producir software de forma organizada, empleando una colección de técnicas y convenciones de notación predefinidas[8]. Ambos conceptos estarán muy relacionados dentro del Proceso Unificado de Desarrollo de Software, el cual indica quién hace qué, cuándo lo hace y cómo lo hace, en otras palabras el Proceso Unificado es la guía para utilizar UML (Unified Modeling Lenguaje). Este capítulo está formado por 5 subtemas, el primero de ellos, el 2.1, presenta la historia del Proceso Unificado, los subtemas 2.2 al 2.5 tienen como objetivo detallar las características, los flujos de trabajo y las fases del Proceso Unificado.

2.1. Historia del Proceso Unificado

El Proceso Unificado (PU) es una resultante de tres décadas de trabajo[9]; su evolución se presenta en la Figura 2.1

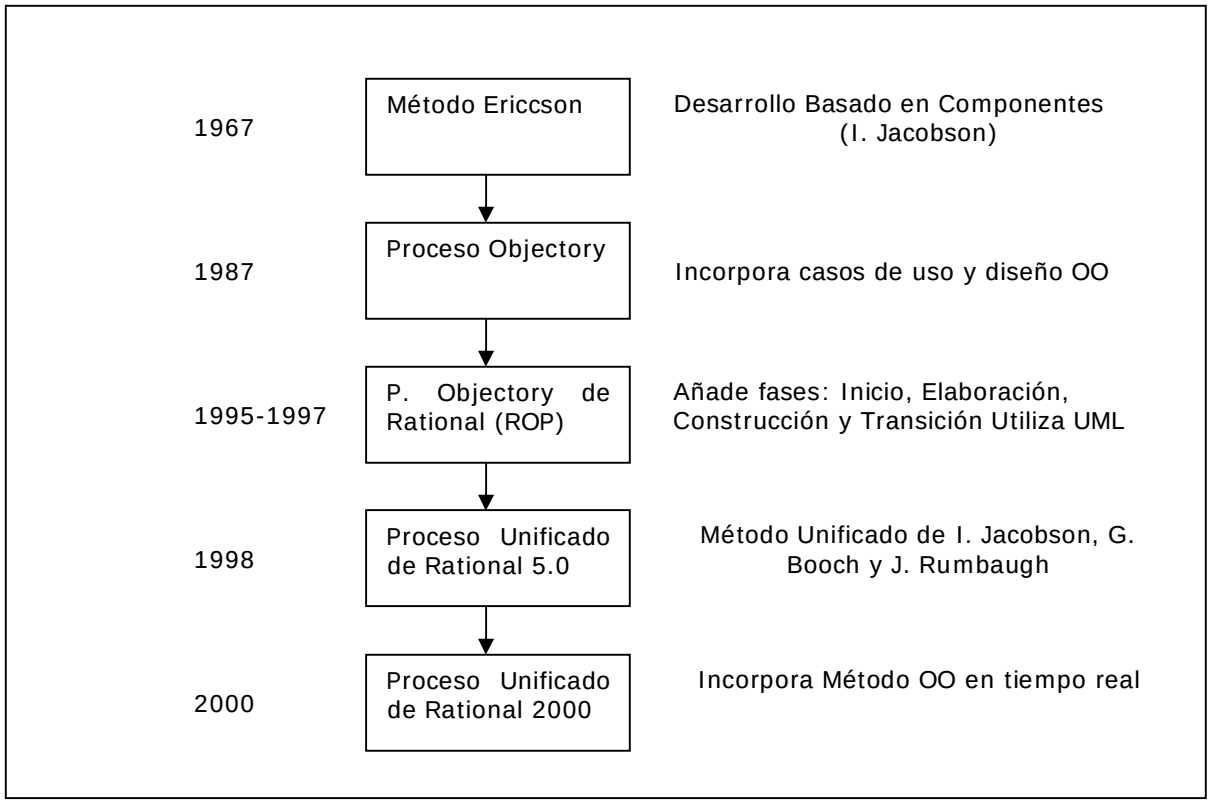


Figura 2.1 Antecedentes del Proceso Unificado
Fuente: Laboratorio de Metodologías y Lenguajes. España.

7 Definición tomada de Software Engineering Body of Knowledge.
8 Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Proceso Unificado de Desarrollo de Software"; Adisson Wesley Iberoamericana, Madrid 1999.
9 Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Proceso Unificado de Desarrollo de Software"; Adisson Wesley Iberoamericana, Madrid 1999.

2.1.1. El método Ericsson

EL método Ericsson fue una de las primeras bases para conformar lo que hoy es el Proceso Unificado. Éste fue creado en 1967 y consistía en representar a un sistema entero como un conjunto de bloques interconectados o pequeños subsistemas; posteriormente se agrupaban los subsistemas de más bajo nivel para convertirlos en un subsistema de más alto nivel. El primer producto del trabajo de las actividades de diseño era una descripción de la arquitectura, que se basaba en una descripción detallada de cada bloque y de sus subsistemas. Un diagrama de bloques describía las interconexiones entre bloques y la forma en que éstos se comunicaban a través de señales o mensajes.

Los mensajes quedaban detallados en una biblioteca de mensajes que era uno de los documentos principales que guiaban el trabajo de desarrollo.

Otro punto importante era la elaboración de diagramas de secuencia, que mostraban cómo los bloques se comunicaban dinámicamente para llevar acabo un determinado requerimiento. La clave de este modelo fue el diseño, a partir de bloques e interfaces bien definidas, lo que permitía utilizar esos componentes en otros sistemas.

En 1976 hubo un avance significativo cuando el Comité Consultivo Internacional Telegráfico y Telefónico (CCITT), el organismo internacional para la estandarización en el área de telecomunicaciones, publicó el lenguaje de especificación y descripción (Specification an Descripción Language, SDL) para el comportamiento funcional de los sistemas de telecomunicación. Dicho estándar especificaba un sistema como un conjunto de bloques interconectados que se comunicaban unos con otros únicamente a través de mensajes. El SDL designaba el término proceso para las clases activas que se encontraban dentro de un bloque. Un proceso poseía instancias de manera muy parecida a como lo hacen las clases en orientación a objetos. Las instancias de los procesos también interactuaban a través de los mensajes. El SDL propuso el uso de diagramas de clases, diagramas de actividad, diagramas de colaboración y diagramas de secuencia. El SDL fue base para lo que hoy es UML.

2.1.2. Proceso Objectory

Ivar Jacobson en 1987 fundó Objetory AB y durante ocho años él y sus colaboradores desarrollaron un proceso llamado Objectory, fusión de las palabras Object Factory (fabrica de objetos). A partir de este proceso fue empleado el término “caso de uso”. Los flujos de trabajo sucesivos se representaron en una serie de modelos: casos de uso, análisis, diseño, implementación y prueba. Cada modelo que se creaba debía estar basado en el modelo anterior y servir de base para la creación del modelo posterior, es decir, debían tener traza entre ellos. La traza entre modelos implicaba también cierto grado de dependencia, pues los cambios a algún modelo o a artefacto dentro de ellos, deberían reflejarse en artefactos o modelos construidos con base al modificado.

Objectory tuvo varias versiones hasta 1985, cada versión nueva se basaba en una anterior utilizando el mismo método Objectory, lo cual era una característica única.

A finales de 1995 La compañía Rational Software Corporation compró a Objectory; con esta fusión Rational logró unir la metodología de Object con características propias que tenían en Rational, como lo eran la abstracción, la ocultación de la información, la reutilización y el prototipado. Las dos contribuciones más importantes de Rational al Proceso fueron el énfasis que puso en la arquitectura y en el desarrollo iterativo.

La arquitectura en Rational consistía de cuatro vistas que representaran al sistema, en vez de tener un solo diagrama que contemplara todo; las vistas eran: la vista lógica, la vista de procesos, la vista física y la vista de desarrollo. Estas vistas permitieron encontrar, tanto a los usuarios como a los desarrolladores, lo que necesitaban para sus diferentes objetivos con la vista adecuada.

2.1.3. El Proceso Objectory de Rational

Con Objectory 3.8 se había demostrado que se podían crear y modelar un proceso de desarrollado de software, ya que se tenía identificado un conjunto de modelos que documentaban el resultado del proyecto. Tenía áreas bien definidas como el modelado de casos de uso, análisis y diseño, pero en otras áreas no estaba bien desarrollado, áreas como son: gestión de requisitos, implementación y pruebas; además de que existían otras de las cuales no se decía mucho, como las de gestión del proyecto, gestión de la configuración, distribución, y obtención del proceso y las herramientas. Es por ello que Rational unió su trabajo con el de Objectory para formar el Proceso de Rational Objectory 4.1, donde se consolidaron las fases de desarrollo y la aproximación iterativa controlada.

Durante el periodo de construcción del Proceso Objectory de Rational (ROP), Grady Booch, (quién tenia su propio método de modelado de objetos, llamado el Método Booch), trabajó con Rumbaugh (que era el desarrollador principal de OMT Object Modelling Technique) en la unificación de los métodos de ambos, lo que dio lugar al Método Unificado 0.8 en Octubre de 1995.

Con la unión de Rational y Objetory AB, Booch, Rumbaugh y Jacobson trabajaron juntos para publicar la versión 0.9 del Lenguaje Unificado de Modelado. Para cuando se liberó la versión 1.1 de UML en 1997, la colaboración no sólo incluía a estos tres personajes, también intervinieron otros metodologistas y empresas de software como IBM, Microsoft y HP, bajo la estandarización del Object Management Group.

2.1.4. El Proceso Unificado de Rational.

Para el año de 1998 EL Proceso Objectory de Rational se había consolidado en un proceso y en junio de ese año Rational publicó una nueva versión del proceso; este nuevo proceso tenía como nombre “Proceso Unificado de Rational 5.0”, dicho nombre reflejó el trabajo que se había logrado al unificar técnicas de desarrollo, a través de UML y la unificación de muchas metodologías.

2.2. El Proceso Unificado

Descripción

El Proceso Unificado (PU) es un proceso de desarrollo de software que está basado en componentes que se conectan a través de interfaces bien definidas, PU utiliza a UML para esquematizar todo el sistema de software, gracias a que PU fue desarrollado paralelamente a UML.

Las tres frases clave que definen a PU son las siguientes:

- Está dirigido por casos de uso.
- Está centrado en la arquitectura.
- Es iterativo e incremental.

A continuación se describen cada uno de los aspectos mencionados.

2.2.1. Dirigido por casos de Uso

Un sistema de software está constituido básicamente por los requerimientos de un usuario, (entendiendo como usuario no sólo a una persona sino también a otros sistemas de software); es por ello que para que un sistema funcione correctamente, se debe haber hecho un correcto análisis de los requisitos del mismo.

Un caso de uso se define como la descripción de un conjunto de secuencias de acciones, incluyendo variaciones que un sistema lleva a cabo, que conduce a un resultado observable de interés para un actor determinado.[10]

Los casos de uso representan los requisitos funcionales de un sistema, entendiéndose por requisitos funcionales todos aquellos que definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Al conjunto de casos de uso se le conoce como modelo de casos de uso, el cual describe en su totalidad la funcionalidad del sistema. Dicho modelo es de gran utilidad ya que es el que guiará la realización del diseño, implementación y prueba; en general, guía el proceso de desarrollo.

Los casos de uso inician el proceso de desarrollo y con esto también se inicia todo el proceso, a través de diversos flujos de trabajo Figura 2.2.

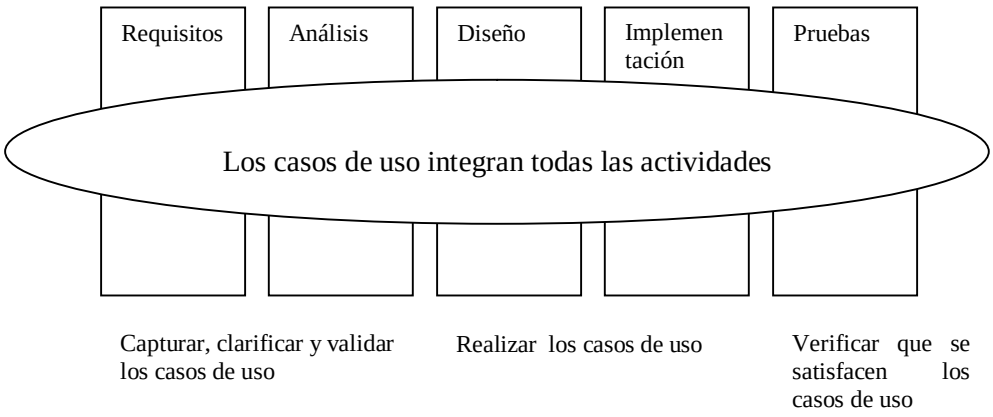


Figura 2.2 RUP es un proceso dirigido por Casos Uso
Fuente: Miguel Ángel Laguna, "2-Proceso", Dpto. de Informática. Universidad de Valladolid, España.

2.2.2. Centrado en la arquitectura

Los casos de uso no se desarrollan aisladamente, éstos son realizados a la par de la arquitectura del sistema. La arquitectura de un sistema es, en su forma mas concreta y de acuerdo a Jacobson, el conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema, las interfaces entre ellos, junto con su comportamiento, las colaboraciones entre estos elementos para formar elementos progresivamente mayores, y el estilo arquitectónico que guía esta organización.

La arquitectura de un sistema se representa mediante vistas del modelo; estas vistas son: una vista del modelo de casos de uso, una vista del modelo de análisis, y una vista del modelo de diseño.

10 Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Proceso Unificado de Desarrollo de Software"; Adisson Wesley Iberoamericana, Madrid 1999.

La arquitectura de un sistema se necesita para:

- Comprender el sistema.
- Organizar el desarrollo.
- Fomentar la reutilización.
- Hacer evolucionar el sistema.

A continuación se explica cada uno de los puntos anteriores.

Comprender el sistema

Para que un sistema funcione correctamente es necesario que todas las personas que intervienen en él, lo conozcan perfectamente. Ésta es una tarea difícil ya que el sistema entre otras cosas posee un comportamiento complejo y opera en entornos complejos, que son tecnológicamente complejos; en algunos casos, como en los sistemas bancarios, son tan grandes, que el proyecto se divide en varios proyectos y a la hora de unirlos hay problemas.

Es por ello que se debe de capacitar a los desarrolladores, directivos, clientes y otros usuarios, para que conozcan lo que se está haciendo. Esto se logra a través del uso de los diferentes diagramas como son los de modelo de casos de uso, modelo de análisis y modelo de diseño. Una vez comprendidos estos diagramas, será más fácil comprender el sistema en su totalidad.

Organizar el desarrollo

Se refiere a una distribución organizada del trabajo; se logra dividiendo un sistema en subsistemas con interfaces claramente definidas.

Fomentar la reutilización

La creación de componentes que puedan ser reutilizados en otros sistemas, es producto de una correcta descripción de la arquitectura.

Hacer evolucionar el sistema

Un sistema debe de ser flexible, es decir debe ser capaz de soportar modificaciones sin sufrir problema alguno.

La arquitectura de un sistema se crea a través de iteraciones, Figura 2.3, durante la fase de elaboración, comenzando por los requisitos y siguiendo con el análisis, diseño, implementación y pruebas. Sólo son considerados los casos de uso más relevantes para la arquitectura, entendiéndose por más relevantes: ***a)*** aquellos que ayudan a mitigar los riesgos más importantes; ***b)*** aquellos que son los más importantes para los usuarios; y ***c)*** aquellos que cubren todas las funcionalidades significativas. Al final de la fase de elaboración se tiene una base de la arquitectura que detalla cómo el sistema debe quedar; se dice que la línea base de la arquitectura es una sistema pequeño y flaco, [11] porque tiene las versiones de todos los modelos que un sistema terminado contiene al final de la fase de construcción. Incluye el mismo esqueleto de subsistemas, componentes y nodos que un sistema definitivo, pero no incluye todo el contenido del esqueleto. El sistema flaco deberá desarrollarse hasta convertirse en un sistema completo, y los cambios que se realicen para lograr esto no deberán de

11 Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Proceso Unificado de Desarrollo de Software"; Adisson Wesley Iberoamericana, Madrid 1999.

afectar la línea base de la arquitectura, ya que está al final de la fase de elaboración debió de ser muy estable.

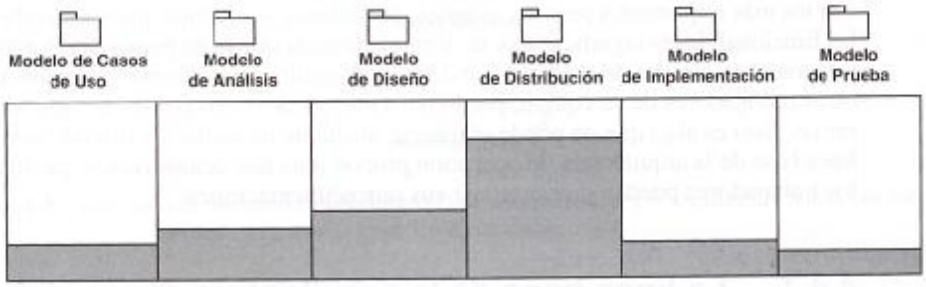


Figura 2.3 La línea base de la arquitectura es una versión interna del sistema, que está centrada en la descripción de la arquitectura.
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

El Proceso Unificado marca una serie de modelos que se desarrollan durante las fases de creación del sistema. Un modelo es un conjunto de artefactos que describen cierto aspecto del sistema, así se tienen los siguientes modelos:

- Modelo del Negocio: establece una abstracción de la organización en donde se utilizará el sistema a desarrollar.
- Modelo del Dominio: establece el contexto del sistema.
- Modelo de Casos de Uso: describe el sistema desde un punto de vista de las funcionalidades para el usuario, de una forma muy simple.
- Modelo del Análisis: donde se refinan y estructuran los requerimientos del sistema.
- Modelo del Diseño: donde se describen los casos de uso desde un punto de vista de la solución de los mismos y tomando en cuenta requerimientos no funcionales.
- Modelo de Procesos: establece los mecanismos de sincronización y concurrencia del sistema.
- Modelo de Despliegue: donde se describe el sistema considerando los nodos que forman la topología de hardware sobre la que se ejecutará el sistema.
- Modelo de Implementación: donde se abarcan los componentes usados para el ensamblado y lanzamiento del sistema ejecutable.
- Modelo de Pruebas: establece las formas de validar y verificar el sistema

La descripción de la arquitectura se obtiene a través de la línea base de la misma, y ésta debe de presentar las vistas de los modelos. La descripción de la arquitectura también detalla los requisitos no funcionales que se especifican como requisitos adicionales, entre ellos están la seguridad, distribución y concurrencia. La descripción de la arquitectura debería de incluir también una descripción de la plataforma en que se está desarrollando, así como el software comercial que se utilizará.

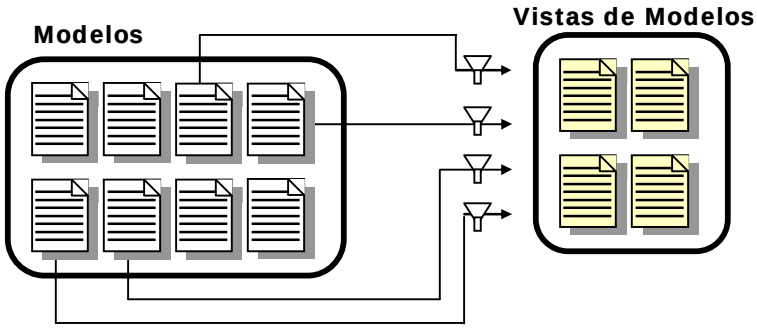


Figura 2.4 Vistas y Modelos.
Fuente: Ing. Luis A. Guzmán "Metodología Para Proyectos De Software Orientado A Objetos Basada En Cmm y El Proceso Unificado De Desarrollo De Software", Tesis Licenciatura, FI - UNAM, 2003.

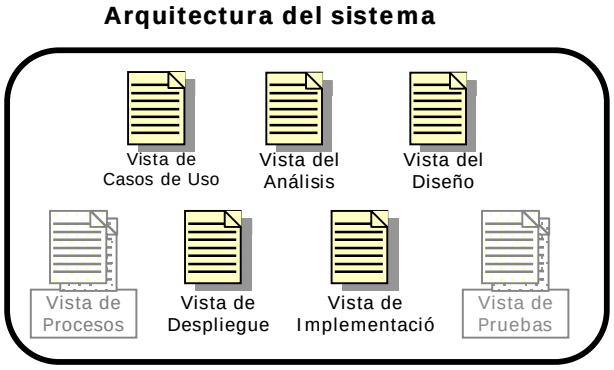


Figura 2.5 Arquitectura y Vistas.
Fuente: Ing. Luis A. Guzmán "Metodología Para Proyectos De Software Orientado A Objetos Basada En Cmm y El Proceso Unificado De Desarrollo De Software", Tesis Licenciatura, FI - UNAM, 2003.

2.2.3. Iterativo e Incremental

Es conveniente dividir un sistema muy grande en partes más pequeñas o mini proyectos, donde los mini proyectos son una iteración que posteriormente resulta en un incremento. Las iteraciones se hacen mediante la selección de un grupo de casos de uso relevantes; posteriormente se crea un diseño utilizando la arquitectura seleccionada como guía, se implementa el diseño mediante componentes y se comprueba que estos componentes satisfacen los casos de uso; si la iteración cumple con los requisitos, se continua con la siguiente .

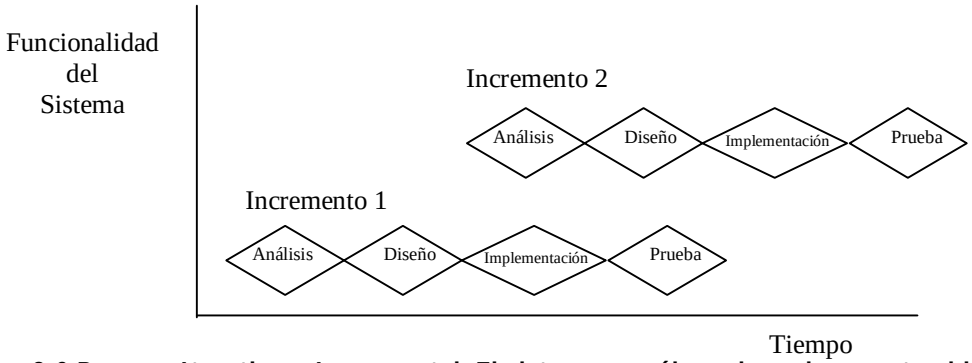


Figura 2.6 Proceso Iterativo e Incremental: El sistema no sólo mejora sino que también crece.
Fuente: Miguel Ángel Laguna, "2-Proceso", Dpto. de Informática. Universidad de Valladolid, España.

A continuación se describe lo que se logra al tener un proceso iteratrivo e incremental:

Se reduce el costo del riesgo.

Al ser una iteración controlada si se tiene que repetir una iteración sólo se perderá el esfuerzo de esa iteración y no de todo el sistema; además es posible empezar a identificar los riesgos desde las primeras iteraciones, de esta manera cuando se alcanza la fase de construcción quedan pocos riesgos importantes (Figura 2.7).

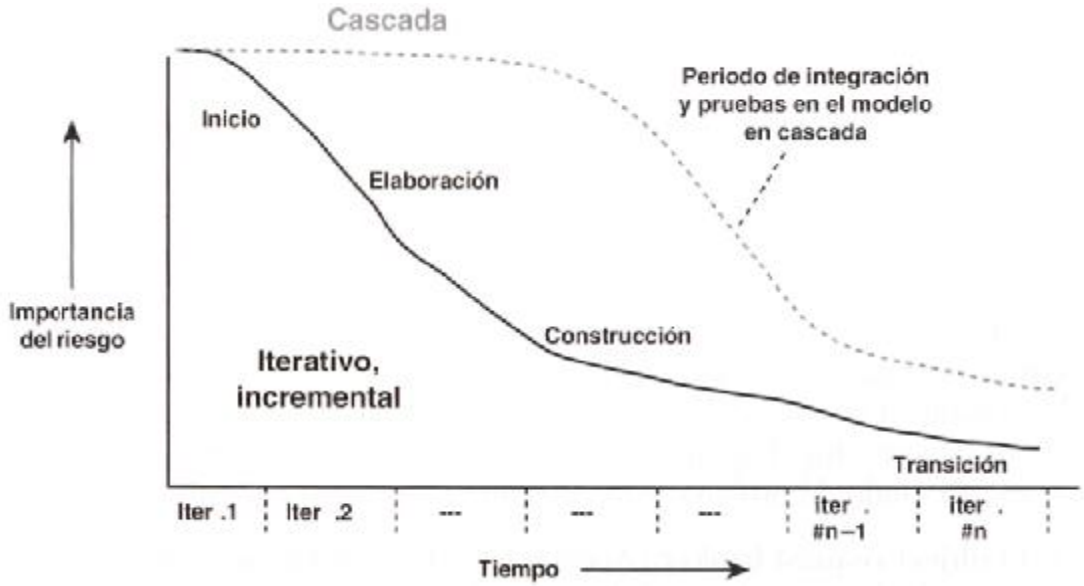


Figura 2.7 Los riesgos importantes se identifican y se reducen al principio del desarrollo iterativo.
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

Obtención de una arquitectura robusta

La arquitectura es resultado de las primeras fases. Debido a que en las primeras fases, el trabajo invertido en la arquitectura no es muy grande, si después de algunas iteraciones se nota que la arquitectura no es la adecuada, se pueden hacer algunos cambios que permitan que los casos de uso se adapten mejor.

Gestión de requisitos cambiantes

Cada iteración del sistema debe de progresar mediante una serie de construcciones, hasta lograr un resultado esperado y por lo tanto un incremento. Esto para el usuario se reduce en entregarle periódicamente pequeñas funcionalidades del sistema, lo que le permite ver de una manera más sencilla a los clientes e identificar nuevas funcionalidades o modificaciones a las que ya se tienen.

Permitir cambios tácticos

Mediante al método iterativo e incremental, los desarrolladores pueden detectar tempranamente problemas y darles solución, evitando que afecten al sistema posteriormente.

Conseguir una integración continua

Al final de cada iteración exitosa, se cumple con un caso de uso que será integrado con otros que se realicen en iteraciones posteriores; de esta manera el sistema va quedando totalmente integrado mostrando el progreso del proyecto.

Conseguir un aprendizaje temprano

Debido a las iteraciones, el equipo de trabajo se va familiarizando con los diferentes flujos de trabajo logrando así una mejor comprensión de cada uno de ellos. En el caso de nuevo personal, es fácil enseñarles ya que trabajarán con alguien que ya tiene perfectamente comprendido como se trabaja dentro de los diferentes flujos.

2.3. La Vida del Proceso Unificado

El ciclo del Proceso Unificado consta de cuatro fases: inicio, elaboración, construcción y transición; cada fase está dividida a su vez en iteraciones. La Figura 2.8 muestra el ciclo de vida de Proceso Unificado.

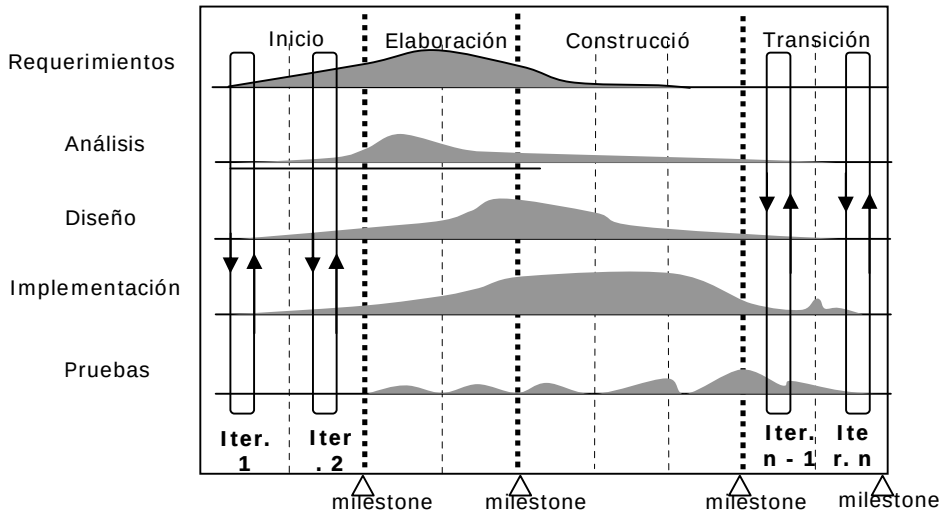


Figura 2.8 Ciclo de vida de RUP
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

En la figura, del lado izquierdo se presentan los flujos de trabajo –requisitos, análisis, diseño, implementación y prueba. Las curvas muestran una aproximación de hasta donde se llevan a cabo los flujos de trabajo dentro de cada fase.

Es importante señalar que dentro de cada fase existen iteraciones que pasan a través de los cinco flujos de trabajo. Estas iteraciones terminan en un hito o piedras pilares (milestone), que son puntos de control en los cuales los participantes en el proyecto revisan el progreso del mismo. Con los hitos se pretende: sincronizar las expectativas y la realidad, identificar los riesgos, y evaluar la situación global del proyecto. También los hitos son necesarios para ver resultados tangibles y compararlos con las expectativas; existen dos niveles de hitos, los principales al final de cada fase y los secundarios al final de cada iteración.

Las fases mostradas en la figura, se explican brevemente a continuación:

Fase de inicio: Al comenzar un proyecto hay que contestar siempre algunas preguntas:
¿Cuál es la visión del sistema?, ¿Es viable?, ¿Se puede comprar o hay que fabricar el sistema?, ¿Cuánto va a costar? Y, finalmente ¿seguimos adelante o paramos? El objetivo de la fase de inicio es desarrollar el análisis de negocio hasta el punto necesario para la puesta en marcha del proyecto; para ello es necesario:

- Delimitar el alcance y objetivos del proyecto.
- Definir la funcionalidad y capacidades del producto.
- Tener una idea de la arquitectura (arquitectura candidata).
- Reducir los riesgos cuanto antes.
- Hacer estimaciones iniciales de costos y tiempos.

Al comienzo de la fase de Inicio, se establece una planificación provisional y los criterios de evaluación de la fase. Al final se debe ser capaz de:

- Fijar el ámbito del sistema.
- Resolver ambigüedades en los requisitos.
- Determinar una arquitectura candidata.
- Mitigar los riesgos críticos.
- Analizar las posibilidades de “negocio” (evaluar el “caso de negocio”).

Fase de elaboración: El objetivo de esta fase es especificar en detalle la arquitectura del sistema. La mayoría de los casos de uso son priorizados y utilizados para la creación de un punto clave dentro de esta fase, que es la creación de la línea base de la arquitectura del sistema, es decir crear una arquitectura estable que guíe el sistema.

Al comienzo de la fase de Elaboración:

- Se planifica la fase y se forma el equipo.
- Se establecen los criterios de evaluación que habrá que cumplir al final:
  - Respecto a los requisitos:
    - ¿Se han identificado? ¿Se han detallado lo suficiente?
  - En cuanto a la arquitectura:
    - ¿Satisface los requisitos? ¿Es robusta?
  - Los riesgos:
    - ¿Se han eliminado los críticos? ¿Se ha completado la lista?
  - Evaluación del proyecto:
    - ¿Se puede fijar un precio y una fecha de entrega?

Fase de Construcción: El propósito de esta fase es completar la funcionalidad del sistema; para ello se debe: ***a)*** clarificar los requerimientos pendientes; ***b)*** administrar el cambio de los artefactos construidos; y ***c)*** ejecutar el plan de administración de recursos y mejoras en el proceso de desarrollo para el proyecto, todo esto para que la línea base de la arquitectura se convierta en la arquitectura definitiva del sistema. Al final de esta fase se tienen todos los casos de uso concluidos, tanto los que son de uso del cliente como aquellos diseñados por el personal; esto no significa que estén libres de errores, ya que la mayoría de ellos serán detectados y solucionados en la fase de implementación.

Fase de Transición: El objetivo de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto al inicio del mismo, para lo que se crea una versión beta del proyecto, que es donde un número reducido de personas realizarán las pruebas e informarán de las deficiencias del mismo.

2.4. FLUJOS DE TRABAJO FUNDAMENALES DEL PROCESO UNIFICADO

PU en su forma genérica, define las siguientes cinco actividades (flujos) a realizar en cada fase del proyecto:

- Captura de requisitos
- Análisis de requisitos
- Diseño
- Implementación
- Prueba

Cada uno de los flujos se realiza por cada iteración que se haga y no sólo una vez en todo el proyecto (Figura 2.9). Es posible que no todos los flujos se utilicen en una fase determinada, como por ejemplo en la fase de inicio no es necesario realizar los flujos de implementación y pruebas.

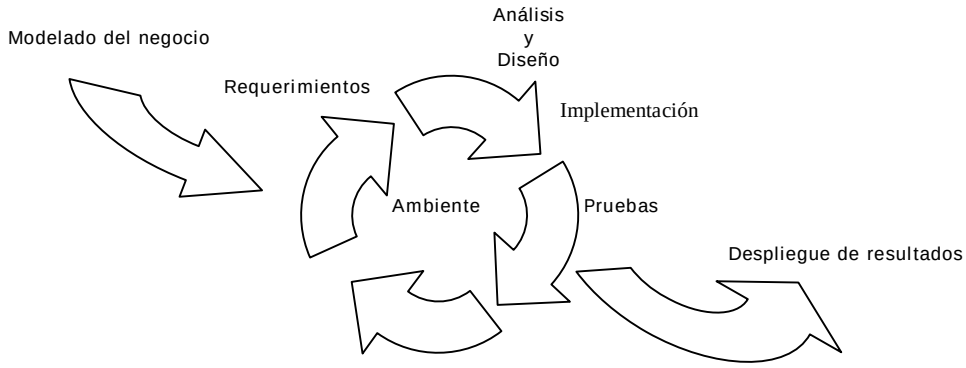


Figura 2.9 Rotación de los flujos
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

Cada flujo de trabajo consta de Actividades, Roles y Artefactos, mostrados como un diagrama de actividades. Los flujos de trabajo producen salidas que sirven de entrada a otra actividad. Cada una de las Actividades es realizada por una persona o personas con un perfil específico al cual se denomina Rol, así un Rol no necesariamente representa a una sola persona.

Los roles dentro del Proceso Unificado están representados por la Figura 2.10

Figura 2.10 Representación de un Rol en PU

Algunos de los roles principales dentro de PU son:

Analista de sistemas: Es quien dirige y coordina el conjunto de requisitos que están modelados en los casos de uso. Es responsable también de delimitar el sistema encontrando los actores y los casos de uso, asegurando que el modelo de casos de uso esté completo; además, debe garantizar la consistencia del sistema, para lo que debe de hacerse cargo del glosario de términos comunes, nociones y conceptos, durante la captura de requisitos.

Especificador de casos de uso: Es quien detalla la especificación de una parte de la funcionalidad del sistema, describiendo los requerimientos de los casos de uso.

Diseñador de interfaz de usuario: Es el encargado de dirigir y coordinar el prototipo y diseño de la interfaz del usuario para los casos de uso; no es el encargado de la implementación real de la interfaz, ya que eso lo hacen los desarrolladores.

Arquitecto: Es el que dirige y coordina las actividades técnicas y los artefactos a través del proyecto. También es el responsable de establecer la arquitectura de las vistas.

Diseñador de pruebas: Es el responsable de definir las pruebas necesarias para garantizar el buen funcionamiento del sistema, lo que involucra la correcta elección de técnicas y herramientas para llevar a cabo las pruebas, así como también para definir y mantener una arquitectura autónoma de pruebas, y para verificar los alcances de las pruebas.

Ingeniero de Pruebas de Integración: Participa en el Flujo de Trabajo de Pruebas. Es responsable de realizar las Pruebas de Integración del Sistema cada vez que se crea una construcción nueva y se agrega al sistema en desarrollo; realiza las Pruebas de Integración con base a los Casos de Prueba. También es responsable de documentar los defectos que encuentre durante la realización de las Pruebas de Integración.

Ingeniero de Pruebas del Sistema: Participa en el Flujo de Trabajo de Pruebas y es responsable de realizar las Pruebas del Sistema necesarias sobre una Construcción, que muestren el resultado de una iteración completa, y de documentar los defectos encontrados. Las Pruebas del Sistema se derivan de los Casos de Prueba. Un Ingeniero de Pruebas del Sistema puede ser cualquiera involucrado en el proyecto, aun sin tener conocimiento del funcionamiento interno del sistema, requiriendo sólo tener conocimiento del comportamiento externo del sistema.

2.4.1. Flujo de Trabajo de Requerimientos

El propósito general del flujo de trabajo de requerimientos es hacer que el sistema se desarrolle correctamente, lo que se consigue mediante una descripción detallada de las capacidades y condiciones con las que debe de cumplir, para así garantizar que exista un entendimiento entre el cliente y los desarrolladores, sobre lo que se debe o no hacer.

Este flujo de trabajo contempla los siguientes pasos:

- Enumerar los requisitos candidatos.
- Comprender el contexto del sistema
- Capturar los requisitos funcionales.
- Capturar los requisitos no funcionales.

Durante este flujo se tienen las actividades que se muestran en la Tabla 2.1 El conjunto de todos los requisitos está formado por los diferentes artefactos que se muestran en la columna derecha. El trabajo a realizar influye en uno o más de estos artefactos.

Trabajo a Realizar	Artefactos Resultantes
Enumerar requisitos candidatos	Lista de características
Comprender el contexto del sistema	Modelo del dominio o del negocio
Capturar los requisitos funcionales	Modelo de casos de uso
Capturar los requisitos no funcionales	Requisitos adicionales o casos de uso concretos

Definen una especificación de requisitos tradicional

Tabla 2.1 El conjunto de todos los requisitos está formado por los diferentes artefactos que se muestran en la columna derecha. El trabajo a realizar influye en uno o más de estos artefactos.

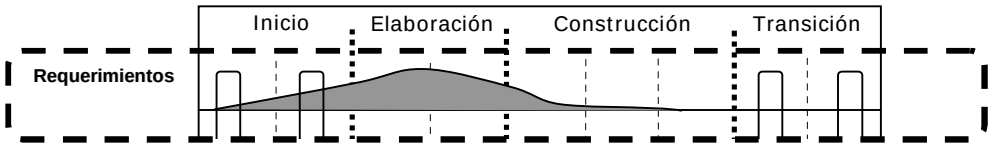


Figura 2.11 El trabajo de los requisitos se hace fundamentalmente durante el inicio y la elaboración
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

La Figura 2.11 muestra cómo el flujo de Requerimientos se desarrolla durante las diferentes fases de PU. En la fase de inicio, los analistas identifican la mayoría de los casos de uso para que de esta forma se conozcan los alcances del sistema y también para conocer cuáles casos de uso son los de mayor relevancia. En la fase de elaboración se deben de haber capturado un 80% de los requisitos y haber descrito la mayoría de los casos de uso. En la fase de construcción se recopilan los requisitos restantes y se implementan. En la fase de transición casi no hay captura de requisitos, a menos que se cambien algunos de los que ya estaban definidos.

A continuación se describen los artefactos generados en el flujo de trabajo de captura de Requisitos.

Modelo de Casos de Uso

Este modelo sirve como un acuerdo entre el cliente y los desarrolladores, ya que describe lo que hace el sistema para cada tipo de usuario, y es de mucha ayuda para las fases de análisis, diseño y pruebas. Un modelo de casos de uso contiene actores, casos de uso y las relaciones que existen entre ellos (Figura 2.12).

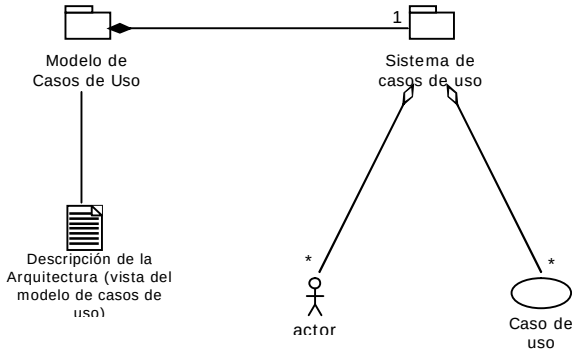


Figura 2.12 El modelo de casos de uso y sus contenidos
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

Actor

Un actor representa a terceros fuera del sistema que colaboran dentro de él; estos actores pueden ser personas físicas u otros sistemas.

Flujo de trabajo

El rol de Analista de Sistemas inicia con la actividad de Encontrar Actores y Casos de Uso para preparar una primera versión del Modelo de Casos de Uso, con los Actores y Casos de Uso identificados. El Analista de Sistemas debe asegurar que el desarrollo del Modelo de Casos Uso captura todos los requerimientos que son entradas del flujo de trabajo, es decir, la Lista de Características y el Modelo del Dominio o del Negocio. Entonces el Arquitecto identificará los Casos de Uso relevantes arquitectónicamente hablando, para proporcionar entradas a la priorización de Casos de Uso (y posiblemente otros requerimientos) que van a ser desarrollados en la iteración actual. Hecho esto, el Especificador de Casos de Uso describe todos los Casos de Uso que se han priorizado. Más o menos en paralelo con ellos, el Diseñador de Interfaz de Usuario sugiere las interfaces de usuario adecuadas para cada Actor, basándose en los Casos de Uso. Entonces el Analista de Sistemas reestructura el Modelo de Casos de Uso, definiendo generalizaciones de funcionalidad entre los Casos de Uso para hacerlo lo más comprensible posible [12](Figura 2.13).

12 Rumbaugh, J.; Jacobson, I.; "El Proceso Unificado de Desarrollo de Software"; Adisson Wesley Iberoamericana, Madrid 1999.

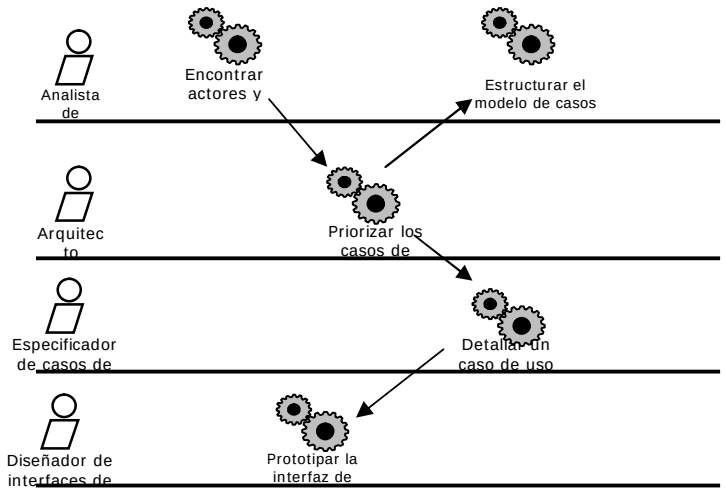


Figura 2.13 Flujo de trabajo del Flujo de Requerimientos
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

2.4.2. Flujo de Trabajo de Análisis

El objetivo de este flujo es analizar refinar y estructurar los requisitos que se describieron en el flujo de requerimientos para que de esta forma se consiga una comprensión más precisa de los mismos y se permita estructurar el sistema completo y también la arquitectura.

En este flujo es importante señalar que, para detallar más los requisitos es posible utilizar el lenguaje propio de los desarrolladores, el cual permite razonar más sobre los aspectos internos del sistema.

El modelo de análisis es un modelo de objetos conceptual que ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema.

La Tabla 2.2 muestra una comparativa entre el modelo de casos de uso y el modelo de análisis.

Modelo de casos de uso	Modelo de análisis
Descrito con el lenguaje del cliente.	Descrito con el lenguaje del desarrollador
Vista externa del sistema.	Vista interna del sistema.
Estructurado por los casos de uso; proporciona la estructura a la vista externa.	Estructurado por clases y paquetes estereotipados; proporciona la estructura a la vista interna.
Utilizado fundamentalmente como contrato entre el cliente y los desarrolladores, sobre qué debería y qué no debería hacer el sistema.	Utilizado fundamentalmente por los desarrolladores para comprender cómo debería darse forma al sistema, es decir, cómo debería ser diseñado e implementado.
Puede contener redundancias, inconsistencias, etc., entre requisitos.	No debería contener redundancias, inconsistencias, etc., entre requisitos
Captura la funcionalidad del sistema, incluida la funcionalidad significativa para la arquitectura.	Esboza cómo llevar a cabo la funcionalidad dentro del sistema, incluida la funcionalidad significativa

	para la arquitectura; sirve como una primera aproximación al diseño.
Define casos de uso que se analizarán con más profundidad en el modelo de análisis	Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

Tabla 2.2 Breve comparación del modelo de casos de uso con el modelo de análisis
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

El papel del análisis dentro de PU se muestra en la Figura 2.14.

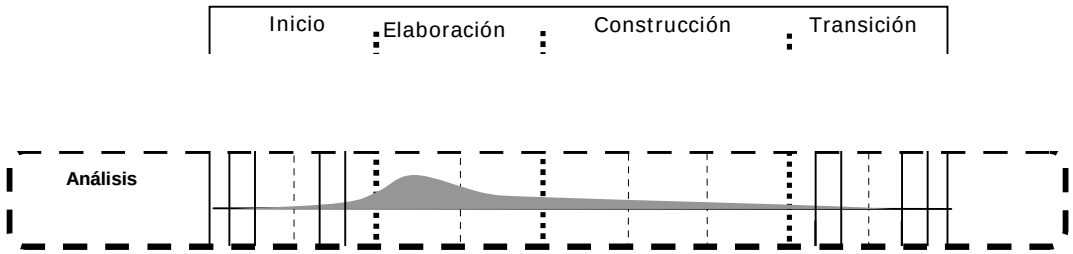


Figura 2.14 Flujo de Análisis.
Fuente: Ivar Jacobson, "El Proceso Unificado de Desarrollo de Software", 1999

En la Figura 2.14 se observa que el flujo del Análisis tiene mayor peso en las primeras iteraciones de la fase de Elaboración, lo que contribuye a obtener una arquitectura sólida y estable y facilita una comprensión en profundidad de los requisitos.

Los artefactos generados en este flujo son los siguientes:

- Modelo de Análisis
- Clase del análisis
- Realización de caso de uso-análisis
- Descripción de la arquitectura (vista del modelo de análisis)

El modelo de análisis se representa mediante un sistema de análisis que denota el paquete de más alto nivel del modelo. La utilización de otros paquetes de análisis es por tanto una forma de organizar el

3. MOPROSOFT

MoProSoft es un Modelo de Procesos para la Industria del Software en México. El principal fin es elevar y garantizar la calidad en las empresas desarrolladoras de software en México. Este modelo de procesos, el cual pretende ser norma mexicana fue impulsado por la Secretaría de Economía, UNAM y AMCIS (Asociación Mexicana para la Calidad de Ingeniería de Software). Tomó como base los estándares de calidad ISO 9000, ISO 15504, ISO 12207, y como marco de referencia los procesos SW-CMM y CMM-I. Antes de ahondar en las prácticas que señala MoProSoft como modelo de procesos, se dará aquí el contexto de este modelo de procesos.

Este capítulo se inicia mencionando aspectos relacionados con la Ingeniería de Software y los procesos de software, pues de esta manera se puede poner en un contexto internacional, las normas y marcos de trabajo que se implementan hoy en día para generar software de calidad. Una vez concluidas estas secciones, se mencionarán los objetivos que persigue, las causas y antecedentes que generaron la creación de MoProSoft y el porqué es equiparable con los estándares y marcos de trabajo mencionados en el párrafo anterior. Para finalizar este capítulo se desarrollarán los puntos centrales que contiene este marco de trabajo para ser implementado en el desarrollo de proyectos de software. Se contemplarán aspectos como su estructura, procesos, tareas, productos, etc.

1.1. Procesos de Software

Hace apenas algunos años, el enfoque que se tenía para la producción de software podría llamarse artístico, sin embargo se han realizado estudios en donde se puede observar la gran cantidad de software fallido o que no cumplía con las expectativas, por lo tanto, con el objetivo de mejorar la calidad se introdujeron nuevos marcos de trabajo para hacer las cosas de manera más eficiente, ayudando a las organizaciones[1] a incrementar la productividad en el desarrollo de software.

El proceso de desarrollo de software es una guía que establece de una forma organizada y administrada la forma de generar software, definiendo en ella actividades, roles, artefactos, metodologías, herramientas utilizadas, flujos de trabajo y las etapas que abarcará el proceso. El proceso de desarrollo en ingeniería está relacionado con el ciclo de vida de un producto de software, su objetivo común es el de lograr un producto que cumpla las expectativas del cliente y además que siga los estándares marcados en la ingeniería de software. Las etapas que enmarcan el ciclo de vida de un producto de software son las siguientes[2]:

- ***Concepción***. Su objetivo es dual: determinar la viabilidad de un proyecto y, en caso de serlo, definir y reunir los elementos materiales y humanos para su desarrollo.
- ***Elaboración***. Su objetivo es definir una serie de fases que incluyen la realización de actividades estratégicas y de planeación con un alto nivel de abstracción.
- ***Construcción***. Su objetivo es construir el producto.

1 Por ***organización*** se hace referencia a instituciones o empresas, completas o una parte de ellas, dedicadas al desarrollo y/o mantenimiento de software.
2 Rumbaugh, J.; Jacobson, I.; Booch, G.; "El Lenguaje Unificado de Modelado, Manual de Referencia"; Pearson Educacion S.A., Madrid 2000

- ***Transición***. Sus objetivos son asegurar que el producto funcione adecuadamente en el ambiente de producción y establecer mecanismos para promover el proceso de mejora continua de desarrollo de software.

Un buen producto de software debe cumplir con las siguientes características: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenimiento y portabilidad. Existe un área que engloba y conjunta teorías, métodos y herramientas para el desarrollo de los productos de software: la Ingeniería de Software. La IEEE (Institute of Electrical and Electronics Engineers) en su estándar 610.12, la define como: "La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software". La ingeniería de software es pues una rama de la ingeniería donde se aplican principios de la computación y las matemáticas en los problemas de desarrollo de software, teniendo en cuenta la relación costo-beneficio.

El SWEBOK (Software Engineering Body Of Knowledge) es un proyecto elaborado por la IEEE para lograr conjuntar o reunir el conocimiento referente siempre a la Ingeniería de Software, organizarlo y dar acceso al mismo; define diez áreas del conocimiento de la ingeniería de software[3]:

- ***Requerimientos de Software***

  Son las necesidades, restricciones y características que deberá satisfacer el producto de software. Debe existir una detallada especificación de los requerimientos, así como la validación de éstos. Una parte importante es saber administrar los cambios que éstos puedan tener durante el desarrollo.

- ***Diseño de Software***

  En esta área se generan los modelos guía para toda la construcción del software; se divide en diseño arquitectónico (referente a la estructura y organización del sistema) y diseño detallado (referente a la descripción de cada componente).

- ***Construcción de software***

  Es el área en donde se crea el software mediante programación, depuración, pruebas unitarias e integración de los diversos componentes. Durante la codificación, se debe buscar reducir la complejidad, optimizar y cumplir estándares de codificación.

- ***Pruebas de Software***

  Consisten en la ejecución del software y la verificación del comportamiento del programa para así compararlo con lo que se esperaba de él, y de esta manera detectar sus fallas. Se realizan durante todo el proceso de desarrollo y por lo general se refinan como se considere necesario. Las pruebas son de diferentes tipos y persiguen diferentes objetivos.

- ***Calidad del Software***

  Es la aplicación de pruebas estáticas e involucra subprocesos referentes al aseguramiento de la calidad como son verificación, validación, revisiones, auditorias, etc.

- ***Mantenimiento de Software***

3 Revista Software Guru, Año 1 No. 1 Enero-Febrero 2005. "Ingeniería de Software. Desarrollar es mucho más que programar". Pág. 42 y 42.

Son las modificaciones al software previamente liberado; hay cuatro (4) tipos de mantenimiento: preventivo, correctivo, perfectivo y adaptativo. Los elementos a considerar en esta etapa son la reingeniería, el análisis de impacto y el "outsourcing".

- ***Administración de la configuración de Software***

  Es la disciplina para identificar la configuración de un sistema, para controlar cambios, mantener integridad, "rastreabilidad", reportar su estado, controlar las versiones, administrar, etc. Estrictamente, por configuración se entiende como el conjunto de elementos de software y hardware que conforman un sistema.

- ***Administración***

  Es la ejecución de tareas administrativas, de una manera temática, disciplinada y cuantificable, para asegurar el desarrollo y mantenimiento del software. Toma en cuenta tareas como la planeación de proyectos, recursos, riesgos, métricas y evaluación.

- ***Proceso***

  Es la definición, implantación, evaluación, mejora y administración del cambio de procesos de las actividades técnicas y administrativas que se realizan para diferentes ámbitos del desarrollo de software.

- ***Herramientas y Métodos***

  Las herramientas sirven para automatizar procesos repetitivos, permitiendo al ingeniero concentrarse en el trabajo creativo de los procesos; los métodos se encargan de sistematizar las actividades de tal forma que aumenten las posibilidades de éxito.

La Ingeniería del Software sigue métodos y tareas para lograr obtener un buen producto de software, las actividades se orientan a la resolución del problema, mientras que los métodos realizan todo un esquema y panorama del problema, desde su formulación y análisis, hasta la búsqueda de soluciones, la elección de la solución más adecuada y la especificación de como llevar a cabo la solución.

Estos modelos pueden clasificarse como muestra la Tabla 3.1[4]:

Modelo	Tipo	Función
Genérico Procesos relacionados con el desarrollo de software.	• **CMM** – Modelo de Madurez de Capacidades • **CMMI** – Modelo CMM integrado • **ISO 9001-2000** – Para la administración de la calidad • **ISO/IEC 15504** – Marco para la evaluación de procesos de software. • **MoProSoft** – Modelo de procesos para la industria mexicana de software.	• Indica qué se debe hacer. • Se debe usar como referencia para definir procesos en una organización y para evaluaciones. • Es un medio para evaluar qué tan bien o mal está una organización.

4 Revista Software Guru, Año 1 No. 1 Enero-Febrero 2005. "Procesos de Software, Diversidad en Modelos".

Modelos específicos Enfocados a la ingeniería de productos de software.	• **PU –** Proceso Unificado • **RUP –** Rational Unified Process • **PSP –** Individuos • **TSP –** Equipos	• Indica cómo se deben hacer las cosas. • Se usan como guía para ejecutar proyectos.

Tabla 3.1 Modelos en la ingeniería de software.

¿Qué estándares son los que deben seguirse para obtener un software de calidad? Un estándar no debe confundirse con las características que debe tener el producto. Los estándares representan acuerdos por consenso, por lo tanto no siempre son ideales. Su mayor valor consiste en la difusión de su terminología, procedimientos, modelos y puntos de referencia. En la **¡Error! No se encuentra el origen de la referencia.** se mencionan los estándares de calidad marcados por la IEEE[5].

Tabla 3.2 Tipos de estándares para Ingeniería de Software.

Tipo	Objetivo
Proceso	Describe mecanismos y conjuntos de actividades relacionados con la ingeniería de productos de software.
Productos de trabajo	Se enfoca en entregables generados por un proceso o un conjunto de procesos o tareas.
Métricas	Define métricas que se usan para medir procesos y/o productos de trabajo.
Formalismos	Define la notación y la representación que es legible para los humanos y máquinas.
Terminología	Define los términos en lenguaje natural utilizados, para los que escriben y usan los estándares.

Estos estándares ayudan a elevar la calidad del producto, vista desde dos puntos diferentes:

- Vista interna, la cual es el grado con el que se cumplen los requerimientos especificados, incluyendo sistemas, componentes o procesos.
- Vista externa, la cual es el grado con el que se cumplen las expectativas del cliente, abarcando el sistema, componentes y procesos.

Cuando se han implantado y definido los procesos de software, se logra: la estandarización de esfuerzos, la consistencia de proyectos, la mejora de prácticas y el entendimiento de las herramientas. Al seguir las prácticas de ingeniería de software se logra lo siguiente:

- Fomenta una repartición del trabajo. Al asignar roles y tareas a los participantes del desarrollo de los procesos.
- Facilita la comunicación de los miembros del desarrollo del proyecto. Debido a que existe una organización y administración confiable y segura del trabajo.

5 Introducción a estándares para la Ingeniería de Software y a SPICE. Estándares de la Ingeniería de Software. 10 de febrero 1999. Hanna Oktaba, Facultad de Ciencias, UNAM. URL: http://hp.fciencias.unam.mx/~ho/SPICE

- Facilita la gestión del proyecto. Dado que se integra el trabajo de cada responsable de las actividades específicas, teniendo un monitoreo y administración permanente del proyecto.

- Permite la reasignación y la reutilización de personal especializado. Una persona se encarga de una o más actividades.

- Permite la transferencia entre proyectos. Las prácticas y actividades pueden ajustarse a nuevos proyectos, sin la necesidad de comenzar "desde el principio".

- Mejora la productividad y el desarrollo. Se obtiene una disciplina y cultura laboral.

1.2. Antecedentes de MoProSoft

En 2002 la Secretaría de Economía (SE) inició el ***Programa para el Desarrollo de la Industria de Software***[6] ***(ProSoft, ver su marco de trabajo en la Figura 3.1)***, que tiene como objetivo fortalecer a la industria de software en México. Cuenta con la participación directa de NAFIN, BANCOMEXT, CONACYT, INEGI, SE, SCT, SECTUR y la Oficina de Políticas Publicas de la Presidencia, a la vez que promueve la incorporación de otras dependencias y entidades públicas relevantes para cada área de trabajo del programa. Es parte del ***Plan Nacional de Desarrollo 2001-2006,*** que plantea el fomento a la industria del software y al rubro de las tecnologías de información, como estrategia para aumentar la competitividad del país.

La SE ha realizado el ***Plan Nacional de Desarrollo*** debido a que:

- México tiene un nivel de gasto en tecnologías de la información y comunicaciones (TIC) de 3.2% del PIB, ubicándose en el lugar 50 a nivel mundial.

- Este rezago es aún mayor en términos de gasto en software, que es 6 veces inferior al promedio mundial y 9 veces menor que el de EUA.

- Países como la India, Irlanda y Singapur han sido exitosos en el desarrollo de su industria de software como motor de su crecimiento económico.

- México cuenta con un gran potencial para desarrollar esta industria dada su cercanía geográfica y el mismo huso horario con el mercado de software más grande del mundo (EUA); su red de tratados comerciales, la más extensa de mundo; y su afinidad con la cultura de negocios occidental.

Se ha detectado que no hay medidas eficientes que fortalezcan la oferta y la demanda, consecutivamente, hay escasas oportunidades para incursionar en el ámbito internacional por parte de las empresas mexicanas. Para lograr el objetivo del desarrollo de las empresas, la SE ha lanzado siete estrategias, más una octava recientemente incorporada, las cuales se listan a continuación:

1. Promover las exportaciones y la atracción de inversiones.

2. Promover la educación y formación de personal competente en el desarrollo de software, en cantidad y calidad convenientes.

3. Contar con un marco legal promotor de la industria.

6 Secretaría de Economía. Programa de Software. URL: http://www.economia.gob.mx/?P=1128

4. Desarrollar el mercado interno.

5. Fortalecer a la industria local.

6. Alcanzar niveles internacionales en capacidad de procesos.

   Se pretende que las empresas cuenten con prácticas internacionales para la producción de software, impulsando la normalización, la investigación y desarrollo, además de la creación de un organismo certificador. Para ello, se hizo una definición de un modelo de procesos y de evaluación, apropiado para la industria de software mexicana.

7. Promover la construcción de infraestructura básica y de telecomunicaciones.

8. Promover código y estándares abiertos

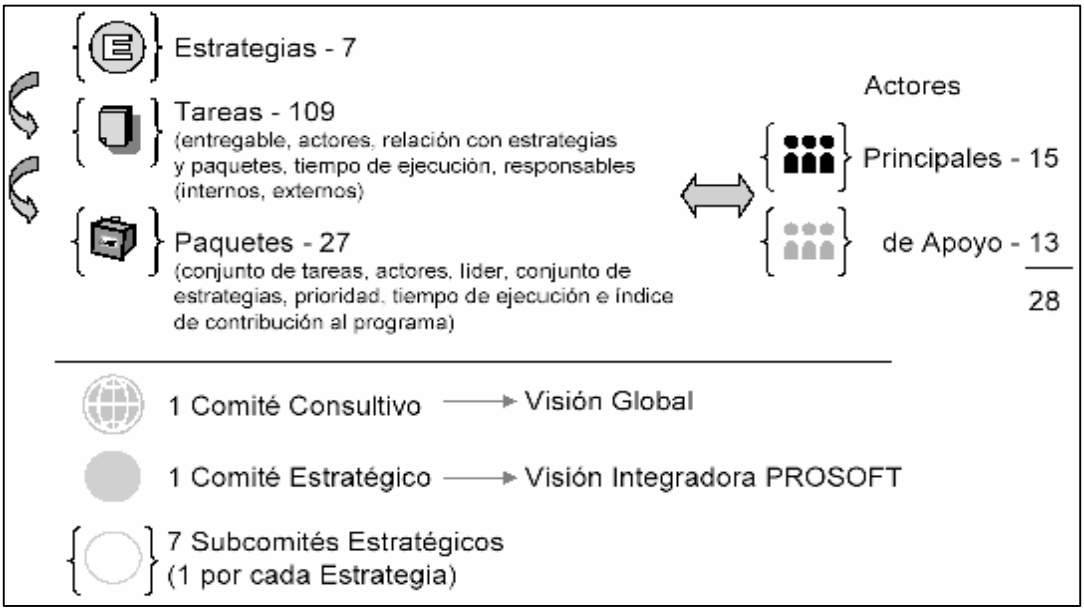


Figura 3.1 Estructura de ProSoft.
Fuente: Secretaría de Economía, "Programa para el Desarrollo de la Industria del Software", Junio 2003

1.2.1. La estrategia 6 de ProSoft

La estrategia 6 de ProSoft enmarca las medidas a tomar para justamente alcanzar niveles internacionales de procesos, tales como han sido planteados por otros modelos como CMM o ISO. Son 7 puntos los que plantea la Estrategia 6, los cuales se listan a continuación:

Estrategia 6 "Alcanzar niveles internacionales en capacidad de procesos".

- Formación de instituciones de capacitación y asesoría en mejora de procesos.
- Definición de modelos de procesos y de evaluación, apropiados para la industria de software mexicana.
- Apoyo financiero para capacitación y certificación de la capacidad de procesos.
- Premio Nacional de Calidad en Tecnologías de Información.

- Estímulos fiscales al desarrollo tecnológico en las empresas.
- Formación de un cajón de financiamiento para actividades de investigación y desarrollo.
- Otros apoyos para actividades de investigación y desarrollo.

De acuerdo al segundo punto, ***definición de modelos de procesos y de evaluación apropiados para la industria del software mexicana***, se evaluaron los modelos ISO 9000, ISO 15504 y SW-CMM, sin embargo ninguno de los estándares que practican éstos eran apropiados para la industria de software mexicana, por lo que se decidió elaborar un modelo basado en los mencionados anteriormente. Para ello, la SE encargó a AMCIS (Asociación Mexicana para la Calidad de Ingeniería de Software) y a la UNAM la elaboración de un ***modelo de procesos para la industria del software***, de ahí el nombre MoProSoft. La primera versión del MoProSoft está por ser liberada y cumple con los requisitos estipulados por el estándar ISO/IEC FDIS 15504-2.

1.2.2. Componentes de la norma mexicana

La Dirección General de Normas (DGN) es quien emite normas para nuestro país a través de NYCE (Normalización y Certificación Electrónica), organismo registrado ante la DGN que está facultado para elaborar, coordinar y emitir Normas Mexicanas de Electrónica, Telecomunicaciones e Informática. Para lanzar la Norma Mexicana de Desarrollo de Software, es requisito indispensable que esté basada en estándares ISO referentes al desarrollo de software y los cuales tienen como marco dos capas principales, un modelo de procesos (ISO 12207) y un modelo de capacidades (ISO 15504).

ISO 15504

Es un marco de trabajo para la evaluación de múltiples modelos, puede ser aplicado a diferentes disciplinas y permite a las comunidades definir sus propios procesos específicos, modelos de referencia y prácticas; maneja el Modelo de Capacidades (Figura 3.2).

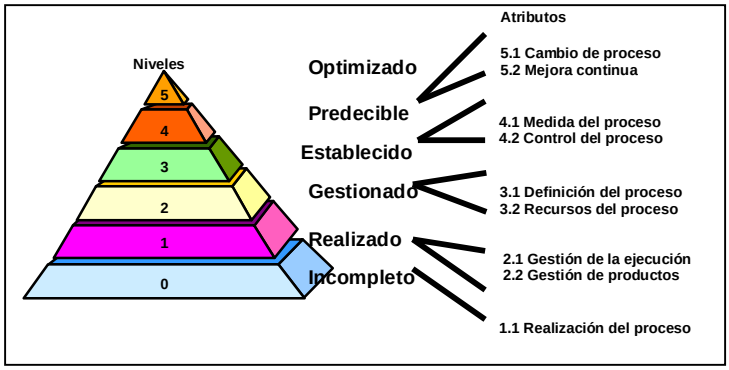


Figura 3.2 Modelo de Capacidades.
Fuente: Dra. Hanna Oktaba, "EvalProSoft y Pruebas Controladas". UNAM, AMCIS

ISO 12207

Presenta una estructura de procesos del ciclo de vida del software, de una forma integral; agrupa los siguientes procesos:

- Implementación del Proceso
- Identificación de la Configuración

- Control de la Configuración
- Contabilidad de Estado de la Configuración
- Evaluación de la Configuración
- Gestión de actualización y distribución

Se propone que la norma mexicana sea dividida en un modelo de procesos, un modelo de capacidades y un método de evaluación, enmarcados dentro de los estándares ISO ya mencionados (Figura 3.3):

Modelo de procesos (MoProSoft)

Contiene a detalle el propósito de cada proceso así como los resultados que se esperan de los mismos (parte normativa), pero también contiene una propuesta de implantación de éstos (parte informativa).

Modelo de capacidades de procesos (qué evaluar)

Se define el método de evaluación; se consideran métricas para llevarlo acabo.

Método de evaluación (cómo evaluar)

Se califican los procesos de tal forma que el resultado indique un nivel que va del 1 al 5, de acuerdo a los resultados obtenidos en las áreas Gestión de Negocio, Gestión de Procesos, Gestión de Proyectos, Gestión de Recursos y Administración de Proyectos Específicos. A la fecha, la SE no ha hecho público en qué consiste (detalladamente) la evaluación mediante EvalProsoft, únicamente, ha dicho a que cualquier empresa que sea evaluada, se le asignará un resultado obtenido en sus procesos.

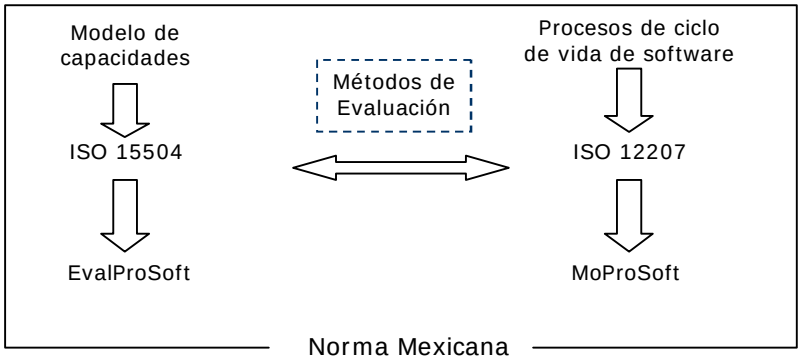


Figura 3.3 Relación entre los estándares ISO 15504 e ISO 12207.
Fuente: Dra. Hanna Oktaba, "Estrategia para la Normalización de la Industria de Software y su Modelo de Procesos (MoProSoft)", Junio 2003.

El objetivo de la norma es permitir lograr una equivalencia entre evaluaciones realizadas con diversos modelos que cumplan la norma 15504 como CMMI, de tal forma que una empresa evaluada mediante la norma mexicana tenga los mismos resultados obtenidos por una empresa evaluada con otro modelo. Para realizar una comparativa, de cómo puede ser equivalente la norma mexicana y un modelo internacional como CMM, se muestran los esquemas en la Figura 3.4, Figura 3.5 y Figura 3.6 [7]:

7 Presentación "Estrategia para la Normalización de la Industria de Software y su Modelo de Procesos (MoProSoft)", Junio 2003, Dra. Hanna Oktaba

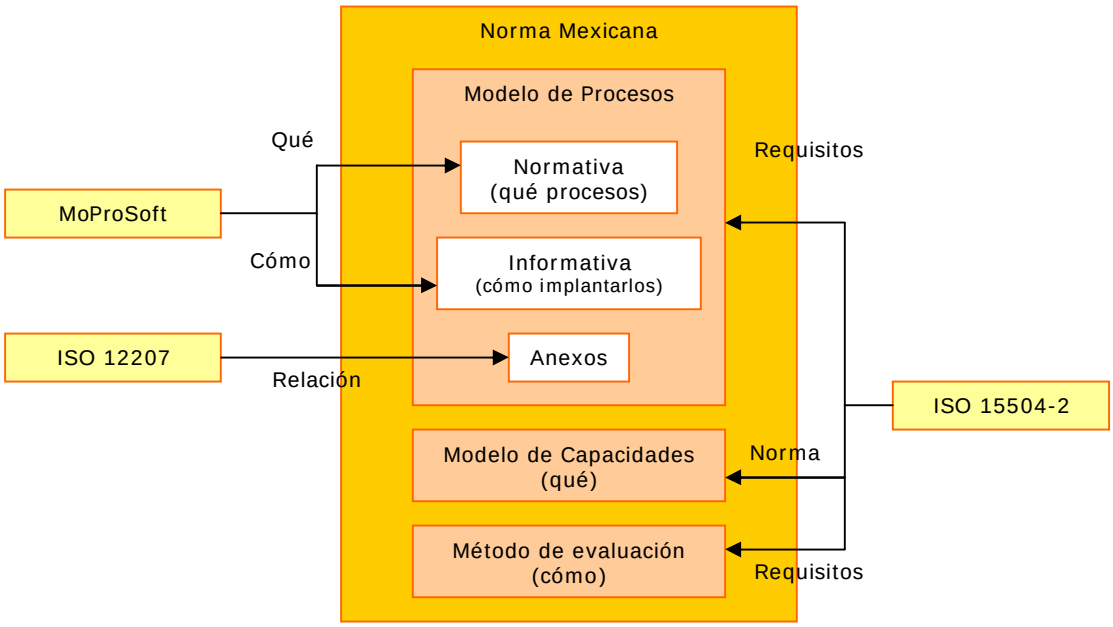


Figura 3.4 Esquema que resume la arquitectura de la norma mexicana.
Fuente: Ana Vázquez "Estrategia propuesta para la emisión de la norma mexicana para la industria de software", AMCIS.

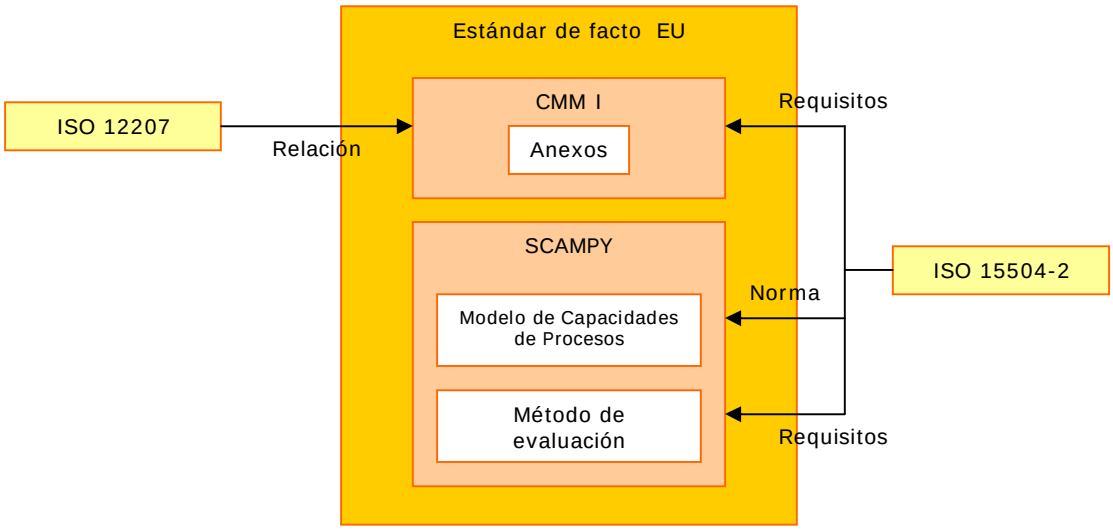


Figura 3.5 Esquema que resume la arquitectura del estándar de facto Estadounidense.
Fuente: Ana Vázquez "Estrategia propuesta para la emisión de la norma mexicana para la industria de software", AMCIS.

La Figura 3.5 muestra la estructura del estándar de facto en CMM, la cual también está basada en estándares ISO. La norma mexicana tendrá como modelo de procesos a MoProSoft, asignando todas las tareas y obteniendo resultados esperados en sus procesos, los que serán evaluados mediante el método EvalProSoft, asignando un cierto nivel de madurez. Así mismo, el marco de referencia de procesos CMM I, el cual maneja entre algunos de sus procesos, la administración de procesos, administración de proyectos y la ingeniería de soporte, también maneja su propio método de

evaluación, llamado SCAMPY, el cual asigna también 5 niveles de capacidad o de madurez a sus procesos.

Ambos modelos (CMM y MoProSoft) por derivarse de los estándares ISO, manejan niveles similares de calidad en sus procesos, ver Figura 3.6. Sin embargo la nueva norma mexicana está diseñada para que tenga un menor costo, un mayor entendimiento y comprensión para las empresas, que el tiempo y recursos empleados sean los justos para implantar la norma, que sea adaptable a cualquier tamaño de empresa y finalmente que sean tan o más competitivas que aquellas certificadas con otros modelos. Por el momento, lo importante para las empresas que se inician en la adopción de modelos de procesos es demostrar no sólo qué están haciendo, sino cómo lo están haciendo.

Se ha propuesto aceptar como válidas las evaluaciones realizadas con métodos y modelos establecidos que cumplan con el estándar ISO 15504, como son CMMI y BOOTSTRAP, sin ningún requisito adicional. Antes de ahondar en los siguientes temas, a continuación se presentan los nombres de las personas que con una ardua labor de trabajo en equipo, han realizado la norma que pretende convertir a México en el principal país desarrollador de software en América Latina:

- ***Hanna Oktaba*** (Director)
- Claudia Alquicira Esquivel
- Angélica Su Ramos
- Alfonso Martínez Martínez
- Gloria Quintanilla Osorio
- Mara Ruvalcaba López
- Francisco López Lira Hinojo
- Maria Elena Rivera López
- Maria Julia Orozco Mendoza
- Yolanda Fernández Ordóñez
- Miguel Ángel Flores Lemus

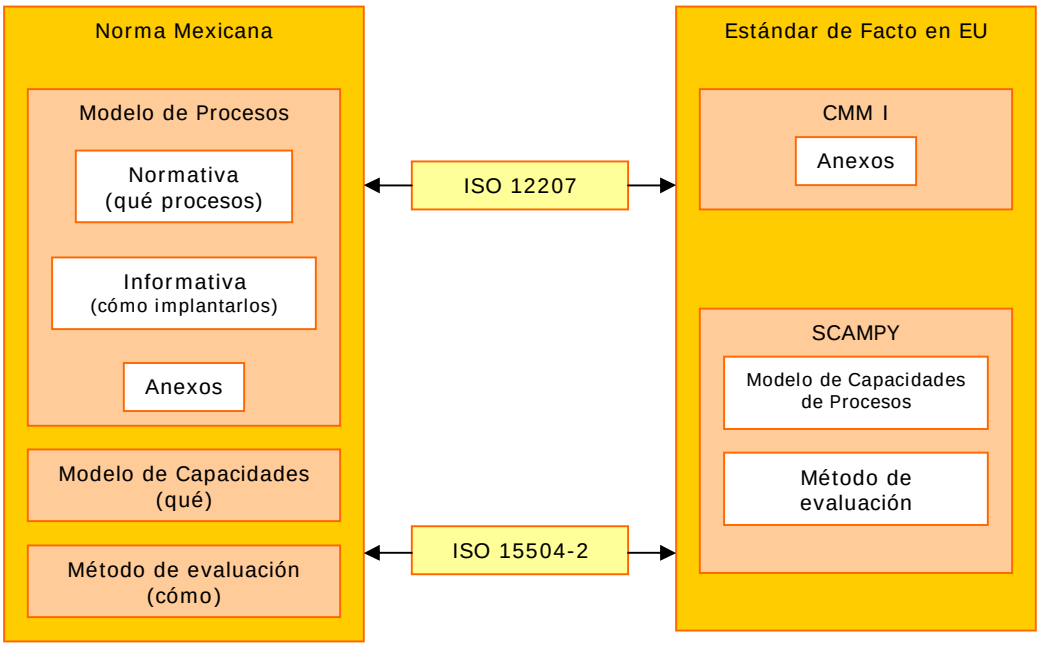


Figura 3.6 La norma mexicana abarca todos los procesos que cualquier otra que esté basada en los estándares ISO 15504; los niveles alcanzados en una u otra norma, son equiparables.
Fuente: Ana Vázquez "Estrategia propuesta para la emisión de la norma mexicana para la industria de software", AMCIS.

1.3. Estructura de MoProSoft

MoProSoft tiene una ***estructura*** que se adapta a las organizaciones, definiendo tres categorías, alta dirección, gestión y operación, las cuales están divididas en seis procesos: gestión de negocio, gestión de procesos, gestión de proyectos, gestión de recursos, administración de proyectos específicos y desarrollo y mantenimiento de software. Además incluye tres subprocesos dentro de la gestión de recursos: recursos humanos y ambiente de trabajo; bienes, servicios e infraestructura, y conocimiento de la organización (Figura 3.7).

Todo el marco de trabajo se basa en productos y roles, donde un rol es el papel que juega una o más personas asignadas de acuerdo a sus habilidades y capacitación para desempeñarlo; las personas deben cumplir con actividades correspondientes a su rol. Un producto es cualquier elemento que se genere por la realización de un proceso.

Para profundizar en cada marco de trabajo de los procesos, se recomienda consultar el documento expedido y disponible en el sitio de la Secretaría de Economía (http:// http://www.economia.gob.mx) ***"Modelo de Procesos para la Industria de Software MoProSoft Versión 1.1 Mayo 2003".***

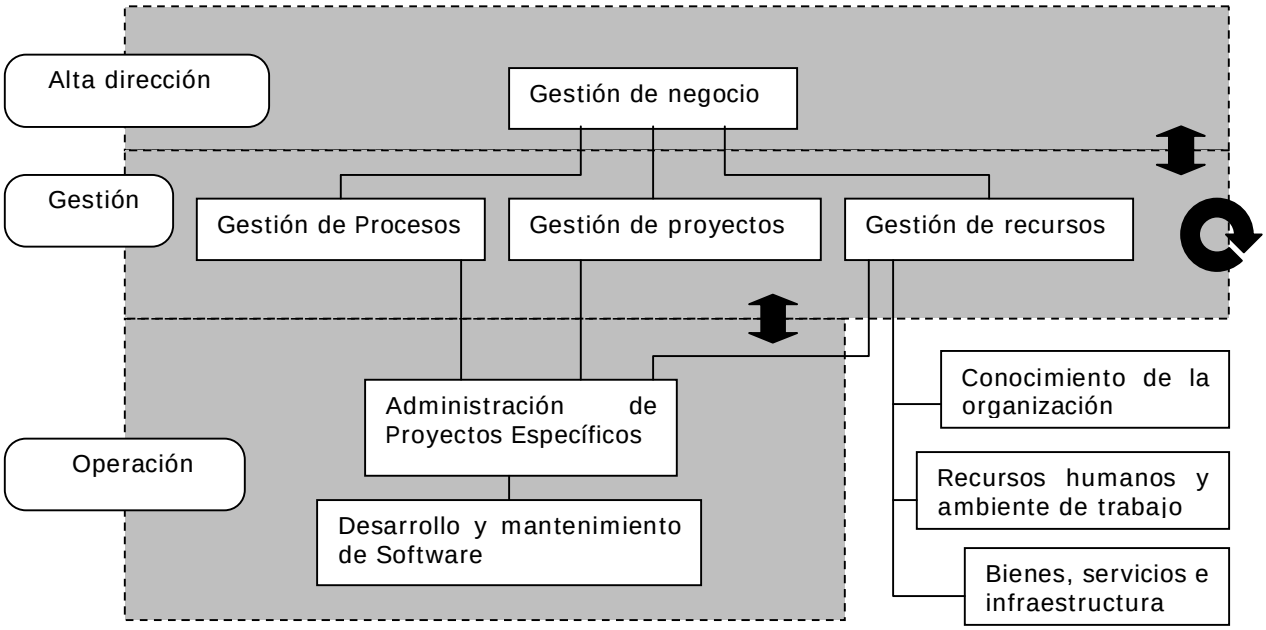


Figura 3.7 Estructura de MoProsoft.
Fuente: Secretaría de Economía, "Modelo de Procesos para la Industria de Software MoProSoft", Versión 1.1, Mayo 2003.

1.3.1. Elementos globales

Todos los procesos siguen un ***patrón de procesos***, el cual es una guía de los elementos que sirven para documentar éstos:

Definición general del proceso

Para definir un proceso dentro de una organización MoProSoft establece los puntos que debe contener la documentación de los procesos seguidos en la organización.

4 APLICACIONES DINÁMICAS DE INTERNET

En los últimos años, ha tomado mayor fuerza e impulso el acceso a conexiones a Internet de alta velocidad, se ha reducido el costo y la infraestructura en los países así lo han permitido; de este modo se abre la posibilidad de manejar recursos más robustos, mejorar la comunicación con dispositivos físicos y potencializar las experiencias de usuario.

Como respuesta a esta oportunidad que apenas comienza en países como México, pero ya con un historial de éxito en otras partes del mundo como EU y Europa, se ha desarrollado una nueva generación de aplicaciones Web llamadas RIA (Rich Internet Applications), el cual puede traducirse como "uso rico de aplicaciones del Internet ó aplicaciones dinámicas de Internet". La finalidad de las RIA es mejorar las experiencias del usuario cuando éste interactúa con aplicaciones Web, pudiendo ser sistemas bancarios, comercio electrónico, revistas electrónicas, etc.

Este capítulo, como su nombre lo indica, centralizará la información en las dos tecnologías principales que constituyen a las RIA: Flash y Flex. Flash se desarrolla en la sección ***tecnología por parte del cliente***, en esta parte se tocarán aspectos relacionados con esta tecnología. Flex se incluye en la sección ***tecnología por parte del servidor***, y se mencionarán aspectos relacionados con la arquitectura y partes que componen a este nuevo lenguaje de programación. Finalmente en la sección de Herramientas de desarrollo se mencionará algo sobre las herramientas de desarrollo empleadas para la construcción del sitio Web.

1.1. Definición

Una aplicación RIA trata de unir lo mejor en cuestiones visuales y tecnológicas, por ejemplo, facilidad de uso, rapidez de despliegue, voz, video, seguridad, etc. De acuerdo con la compañía Macromedia, impulsora y pionera de este tipo de tecnología, recientemente adquirida por la también compañía de software Adobe, para desarrollar una aplicación RIA se debe contar con lo siguiente[1]:

- Rich Client Technology (RCT): proporciona las capacidades, del lado del cliente, que hacen a las RIA posibles, aprovechando el poder de procesamiento local, es decir de los recursos y de las computadoras personales. Los dos factores que eligen la RCT son la adopción de la tecnología y las capacidades con que se cuenta.

- Server Technology: Macromedia Studio MX 2004 integra el ambiente del server-scripting ColdFusion, además de marcos de trabajo en .NET y J2EE con servicios Web o XML. Flash Remoting está disponible para conexiones del lado del servidor, donde se requiere una integración de datos, contando con una tecnología de comunicación de dos vías e intercambio de datos en tiempo real.

- Herramientas de desarrollo: teniendo una tecnología cliente-servidor se debe tener un sistema de herramientas de desarrollo fácil y de gran alcance, que permitan un comienzo rápido y ofrezcan soluciones avanzadas.

1.2. Construyendo una RIA

1 Macromedia. "Developing Rich Internet Applications with Macromedia MX 2004". August 2003

Bajo el nuevo concepto de desarrollo, RIA, se ha implementado una parte del sitio Web del LTST. A continuación se describirán los elementos involucrados.

1.2.1. La tecnología por parte del cliente

Macromedia Flash

Macromedia Flash MX es un software especializado para diseñadores y desarrolladores, el cual permite crear todo tipo de animaciones integradas por audio, video y texto, las cuales pueden llevar a nuevas experiencias dinámicas para los usuarios, dándoles uso en interfaces de aplicaciones, presentaciones interactivas y ahora, aplicaciones RIA. Flash es la plataforma de software de predominancia indiscutible, usada por más de un millón de profesionales y con una presencia en más del 97% de los equipos de escritorio con conexión a Internet en todo el mundo, así como en una amplia gama de dispositivos.

Los archivos que maneja Macromedia Flash, son archivos con extensión "fla". Se puede decir, que los archivos ***fla*** son el archivo fuente de cualquier animación hecha con las herramientas de trabajo de Macromedia Flash, los archivos ***fla*** incluyen el código Action Script, el cual es el lenguaje de programación que utiliza Flash.

Es necesario usar un reproductor llamado Flash Player para que una computadora sea capaz de visualizar las animaciones creadas con Flash, éste también es proporcionado por Macromedia y puede ser recargable en su sitio de Internet, sin embargo el reproductor no interpreta los archivos ***fla***, sino solamente aquellos los archivos compilados, los que tienen la extensión "swf" y son generados dentro del mismo entorno de trabajo de Macromedia Flash. Un archivo ***swf*** puede ser visualizado en los navegadores de Internet, o en reproductores capaces de interpretar esta clase de archivos.

Algunos puntos que hacen interesante y atractivo el uso de Macromedia Flash son los siguientes[2]:

- Creación de experiencias dinámicas e interactivas con el usuario.
- El contenido del archivo ***swf***, generado por el motor en tiempo de ejecución de Macromedia Flash Player, puede ser ejecutado en sistemas operativos Windows, Macintosh y Unix, y en el Web, en PDA y hasta en teléfonos móviles. La versión 7 del Flash player incluye un nuevo modelo orientado a objetos de gran alcance, así como una nueva API para el desarrollo de sistemas client/server.
- Los nuevos componentes para el manejo de datos en Flash MX 2004, permiten trabajar datos que provienen de fuentes externas tales como XML y SOAP para servicios Web.
- Flash soporta Macromedia Flash Remoting, una tecnología que permite la conectividad de alto rendimiento y diversas tecnologías de servidores. Utiliza un formato binario de mensajes, llamado Action Message Format (AMF), que le permite invocar objetos por parte del servidor con una sola línea del código.
- La visualización en pantalla es de alta calidad puesto que ésta es basada en vectores y es ajustable a escala.
- ActionScript 2.0, este lenguaje de programación se ha enfocado hacia el manejo de objetos, esto facilita a los programadores con experiencia en este tipo de programación, por ejemplo a los programadores Java.

1.2.2. La tecnología por parte del servidor

Macromedia Flex 1.5

2 Información resumida del documento "Developing Rich Internet Applications with Macromedia MX 2004". August 2003

Además de las características señaladas de Macromedia Flash, Flex puede crear aplicaciones aún más robustas. El desarrollador Flex se enfrentará al aprendizaje de este nuevo lenguaje de programación, éste se aminora si tiene experiencia en programación particularmente en el lenguaje Action Script 2.0.

Macromedia Flex es un servidor de presentaciones que responde a peticiones de archivos ***mxml***, éstos están instalados en el servidor y son creados por el desarrollador. El servidor Flex responde al usuario compilando los archivos ***mxml***, cuya interfaz son archivos de tipo ***swf*** que el cliente visualiza.

Extraídos del documento denominado "La solución de niveles de presentación para entregar aplicaciones dinámicas de Internet empresariales"[3], a continuación se listan los aspectos relevantes de Flex:

- El servidor de presentación Flex reside en el nivel de presentación del modelo de aplicaciones de n-niveles de una organización.

- Flex le permite a los desarrolladores entregar aplicaciones que le proporcionan a los usuarios una respuesta inmediata, una transición perfecta entre un estado y otro, y entre una pantalla y otra, y un flujo de trabajo ininterrumpido.

- El patrón fundamental es el mismo: se crea un archivo de texto que contiene el código fuente de la aplicación, se despliega el archivo en el servidor, y el servidor compila el código en una aplicación al recibir la primera solicitud, con las solicitudes subsiguientes servidas desde la memoria caché. El servidor de presentación Flex emite una interfaz del usuario para el cliente dinámico, que se ejecuta en el Flash Player.

- La capacidad de integración de Flex facilita el aprovechamiento de código e información existentes usando los servicios Web, acceso a los objetos Java o XML. Flex también se integra con las tecnologías y estructuras de presentación existentes, tales como JSP y Struts.

- El despliegue de la aplicación Flex en la plataforma J2EE se realiza mediante archivos Web Java (WAR). Por lo que puede ser integrado al entorno J2EE, inclusive dentro de una aplicación ya existente.

- Mientras se ejecuta en el Flash Player, la aplicación Flex puede interactuar con la funcionalidad del lado del servidor, tal como los objetos Java, los servicios Web SOAP y otros servicios del lado del servidor.

- La arquitectura de Flex es la que muestra Figura 4.1 Arquitectura del lenguaje Flex.

3 Macromedia® Flex™. "La solución de niveles de presentación para entregar aplicaciones dinámicas de Internet empresariales". Octubre 2004.

5. JAVA: STRUTS

Este capítulo describe el patrón de diseño MVC (Model View Controller) y el framework de Struts, que facilita el desarrollo de aplicaciones Web en Java basadas en dicho patrón. Esta descripción será hecha de manera muy breve, ya que el propósito de este trabajo no es el uso de estas tecnologías.

La utilización de Struts en el desarrollo de aplicaciones Web, garantiza una arquitectura completamente estructurada que divide perfectamente la lógica de negocio (Model), presentación (View) y control de flujo de aplicaciones (Controller). En muchos desarrollos Web se diseña consciente o inconscientemente siguiendo este patrón, por tanto la adopción del modelo Struts no debiera ser complicado.

Para conocer lo que es el framework de Struts en java, es necesario primeramente describir lo que es el patrón de arquitectura MVC (Model-View-Controller), posteriormente en este capítulo se dealla de manera general la arquitectura de Struts así como los componentes que lo conforman, finalmente se muesra un sencillo ejemplo que muesra el funcionamiento de Struts.

5.1. MVC: Model View Controller

MVC o Model View Controller es un patrón de diseño aportado originariamente por el lenguaje SmallTalk,[1] a la Ingeniería del Software. El paradigma MVC consiste en dividir las aplicaciones en tres partes: controlador, modelo y vistas.

En el patrón de diseño MVC, el flujo de la aplicación está dirigido por un Controlador central. El Controlador delega solicitudes a un manejador apropiado. Los manejadores están unidos a un Modelo, y cada manejador actúa como un adaptador entre la solicitud y el Modelo. El Modelo representa, o encapsula, un estado o lógica de negocio de la aplicación. Luego, el control normalmente es devuelto a través del Controlador hacia la Vista apropiada. El reenvío puede ser determinado mediante la consulta a los conjuntos de mapeos, normalmente cargados desde una base de datos o un fichero de configuración. Esto proporciona un acoplamiento cercano entre la Vista y el Modelo, que puede hacer las aplicaciones significativamente más fáciles de crear y de mantener (ver Figura 5.1).

1 Smalltalk es un lenguaje orientado a objetos diseñado por Alan Kay

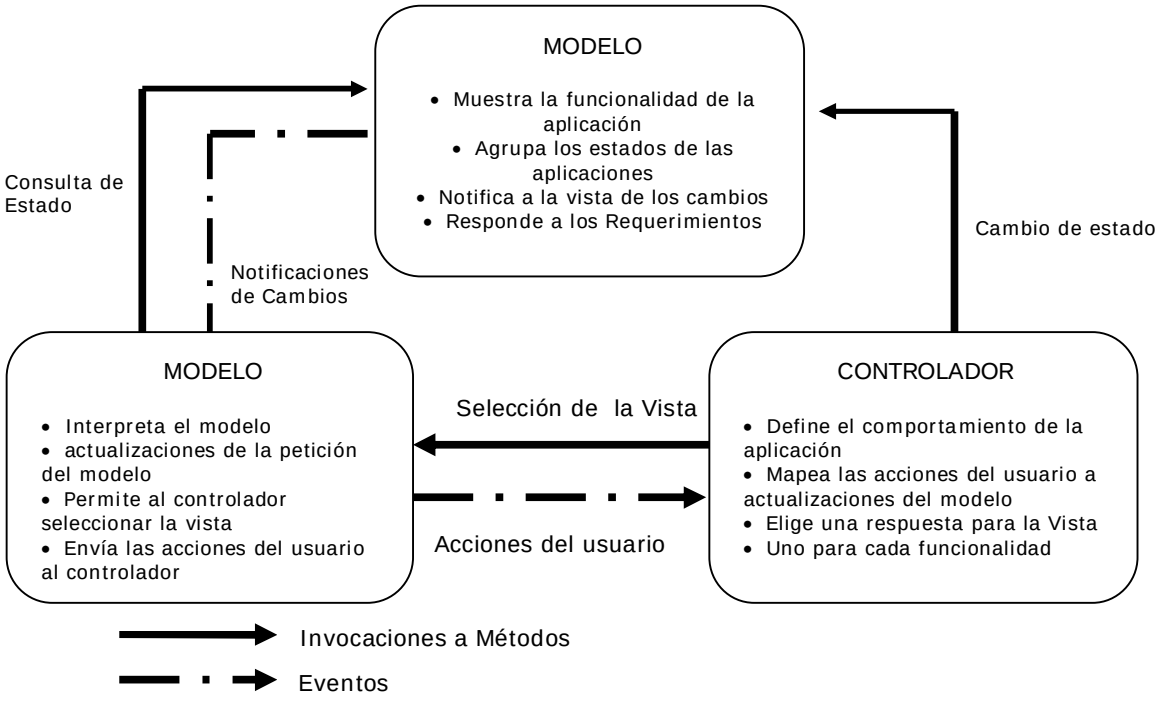


Figura 5.1 Esquema del MVC.
Fuente: José A. Urzúa, "Diseño e Implementación de un Nuevo Sistema de Facturación para NIC". Chile.

A continuación se describe brevemente las partes que integran el modelo MVC

Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista (View): Muestra la información al usuario y obtiene los datos del modelo. Pueden existir múltiples vistas del modelo; cada vista tiene asociado un componente controlador.

Controlador (Controller): La parte Controlador de la aplicación está enfocada en las solicitudes recibidas desde el cliente (normalmente un usuario ejecutando un navegador Web), decidiendo el Controlador qué función de la lógica de negocio se va a realizar, y luego delegando la responsabilidad para producir la siguiente fase de la interfaz de usuario, en un componente Vista apropiado.

El MVC es un patrón ampliamente utilizado en múltiples plataformas y lenguajes. Algunos de sus principales beneficios son:

- Menor acoplamiento
    - Desacopla las vistas de los modelos.
    - Desacopla los modelos de datos.

- Mayor cohesión

    - Cada elemento del patrón está altamente especializado en su tarea (la vista, en mostrar datos al usuario; el controlador en las entradas y, el modelo en su objetivo de negocio).

- Las vistas proveen mayor flexibilidad y agilidad

    - Es posible crear múltiples vistas de un modelo.
    - Es posible crear, añadir, modificar y eliminar nuevas vistas, dinámicamente.
    - Las vistas pueden ser anidadas.
    - Es posible cambiar el modo en que una vista responde al usuario, sin cambiar su representación visual.
    - Es posible sincronizar las vistas.
    - Las vistas pueden ser concentradas en diferentes aspectos del modelo.

- Mayor facilidad para el desarrollo de clientes ricos, en múltiples dispositivos y canales

    - Una vista para cada dispositivo, que puede variar según sus capacidades.
    - Una vista para la Web y otra para aplicaciones de escritorio.

- Más claridad de diseño

- Facilidad en el mantenimiento

- Mayor escalabilidad

5.2. Struts

Struts es un framework para aplicaciones Web java, que implementa el modelo MVC. Realmente lo que provee es un conjunto de clases y TAG-LIBS (Bibliotecas de etiquetas JSP) que conforman el Controlador y la integración con el Modelo (o lógica de negocio), y facilitan la construcción de vistas.

Naturalmente, el Modelo o lógica de negocio es la parte que corresponde desarrollar. Struts es una plataforma sobre la que es posible montar la lógica de negocio, y permite dividir la lógica de la presentación, entre otras cosas.

El marco de trabajo Struts ofrece los siguientes servicios, de acuerdo a los requerimientos de una aplicación Web:

- Un servlet que actúa como controlador central.

- Bibliotecas de etiquetas JSP para la administración de JavaBeans (clases donde se almacenan los datos que proporciona un usuario desde la interfaz), la generación de HTML, el manejo de plantillas y el control de flujo en JSP.

- Una estructura de internacionalización de mensajes, lo que significa que cualquier mensaje que aparezca al utilizar la aplicación se encuentra en el idioma del usuario. Para ello se necesario crear un archivo de recursos de aplicación en el que se incluyan los mensajes adecuados para cada idioma.

- La implementación de JDBC, para definir las fuentes de datos y una agrupación de conexiones a bases de datos.

- Un mecanismo general de resolución de errores y excepciones, lo que implica la recuperación de mensajes de error desde un archivo de recursos de aplicación.

- El análisis de sintaxis (“parseo”) XML.

- Utilerías para cargar archivos.

- Utilerías de conexión.

5.2.1. Arquitectura de Struts

La Figura 5.2 muestra la arquitectura de Struts basada en el modelo MVC.

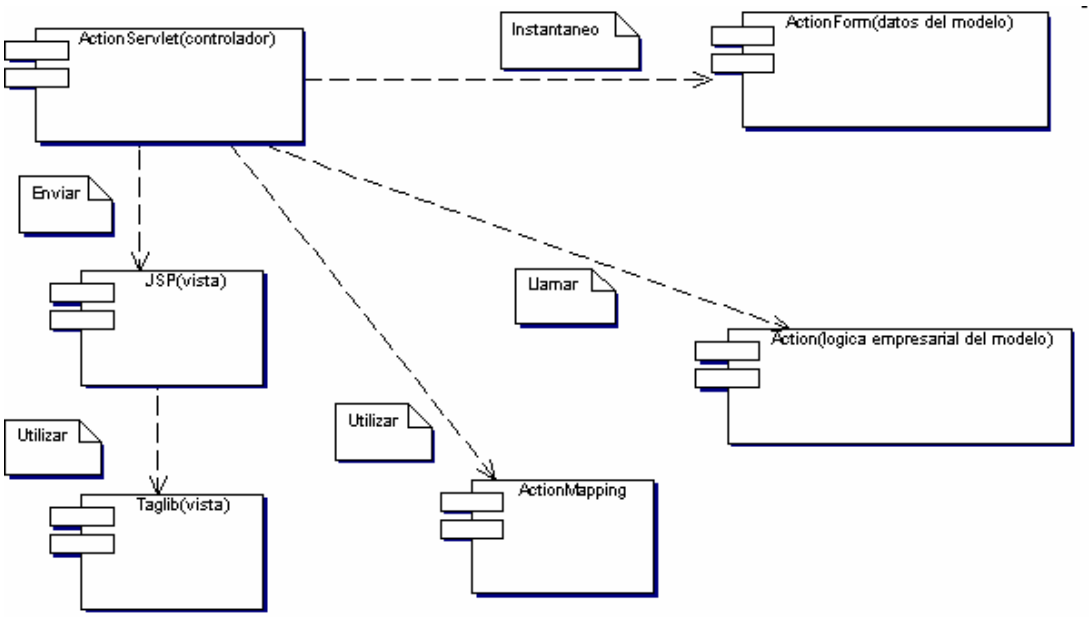


Figura 5.2 Arquitectura de Struts
Fuente: Marcos O. Vázquez, "Aplicación de patrones basados en J2EE para el diseño e implementación de la capa de control de la herramienta integral para MoProSoft (HIM)". México 2005.

Para desarrollar el nivel de presentación (vista) de una aplicación basada en Struts, se utilizan las bibliotecas de etiquetas de Struts (taglib). Todas las solicitudes del cliente se transmiten a un servlet denominado ActionServlet, que actúa como controlador central (este controlador es la implementación del patrón Front Controller[2]). ActionServlet a su vez, pasa los datos de la solicitud a un JavaBean conocido como ActionForm.

Un ActionForm es un JavaBean que representa los datos que son recolectados desde un formulario (datos que envía el usuario desde una interfaz). Estos formularios son generados por las páginas JSP, con ayuda de la biblioteca de etiquetas html de Struts. El ActionForm tiene la posibilidad de validar los datos (implementación del patrón Validator) antes de pasarlos al ActionServlet para su posterior procesamiento.

Un ActionMapping es un objeto que permite redireccionar la solicitud del cliente, es decir, si la solicitud es exitosa, la petición se continúa procesando hasta ejecutar cierta funcionalidad, pero si la solicitud es errónea, la petición se procesa de nuevo solicitando una vez más los datos.

Las propiedades más importantes de ActionMapping son:

- type: Nombre totalmente cualificado de la clase Java, que implementa la clase Action usada por este mapeo.
- name: El nombre del bean de formulario, definido en el archivo de configuración que usará esta action.

[2] Un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. El patrón Front Controller podría dividir la funcionalidad en 2 diferentes objetos: el Front Controller y el Dispatcher. En ese caso, El Front Controller acepta todos los requerimientos de un cliente y realiza la autenticación, y el Dispatcher direcciona los requerimientos a manejadores apropiada.

- path: El path del requerimiento solicitado, que corresponden con la selección de este mapeo.

- forward: El path del requerimiento solicitado, a la que se pasa el control cuando se ha invocado su mapeo.

Estas propiedades son guardadas en un archivo llamado struts-config.xml, como se describirá mas adelante.

El objetivo de las clases Action es el de procesar una solicitud y devolver un objeto que identifica dónde se debería reenviar el control, para proporcionar una respuesta adecuada.

5.2.2. Configuración de Struts

Como ya fue mencionado, ActionServlet es el principal componente del elemento controlador, es el responsable de la delegación de las solicitudes. Para lograr esto, se necesita un tipo de guía que dirija las solicitudes a los componentes correspondientes, es decir, que comprenda desde la asignación de una determinada solicitud (basada en su dirección URL), hasta el componente que procesará dicha solicitud. Toda esta información acerca de la asignación, es guardada en objetos ActionMapping. Estos objetos se configuran mediante un archivo XML, estos objetos son creados por el programador de la aplicación.

El archivo XML, comúnmente conocido como struts-config.xml, contiene la siguiente información:

- La definición de los elementos JavaBeans que almacenarán los datos recogidos desde la interfaz. Cada elemento que representa un JavaBean contiene:
    - Un nombre.
    - La ruta de la clase que almacenará los datos.

- La definición de los elementos ActionMapping, que sirven para configurar los objetos ActionMapping. Cada elemento ActionMapping contiene:
    - El nombre del servlet que atenderá la petición.
    - La ruta de la clase que implementará al servlet. Esta clase es una clase Action que implementa la lógica de la aplicación.
    - El nombre del JavaBean que utilizará la clase Action para procesar los datos que proporcione un usuario.
    - El alcance del JavaBean.
    - La ruta del JSP, en el caso que se presente un error en la información que proporciona el usuario desde la interfaz.
    - El valor de validación del JavaBean. Esto se aplica cuando se quiere validar la información del JavaBean antes de que sea procesada por la clase Action.
    - La ruta del recurso al que se dirigirá la respuesta, en el caso de que el procesamiento haya sido exitoso.

- La definición de fuentes de datos JDBC para el acceso a bases de datos. Aquí se define el nombre del controlador de dicha fuente de datos, la ruta de la fuente de datos, el nombre y contraseña de acceso a la fuente de datos, y el número de conexiones.

5.2.3. Componentes de Struts

Los componentes de Strus son los siguientes: El controlador ActionServlet, la lcase Action, el bean ActionForm, el objeto ActionMapping, La bliblioteca de etiquetas de Sruts. A continuación se describen cada de uno de estos componentes.

<u>El controlador ActionServlet</u>

1. DESARROLLO DEL SITIO WEB LTST

En el presente capítulo se explica mediante la metodología enmarcada por ***MoProSoft***, cómo fue construyéndose el sitio Web LTST, no de una forma tradicional, sino enriqueciéndolo con la nueva propuesta de desarrollo vía Web, es decir, bajo las RIA, las cuales ya fueron explicadas en el Capítulo 5.

MoProSoft está orientado para que una organización alcance ciertos objetivos o metas, bajo un esquema de trabajo establecido. ***MoProSoft*** contempla 5 ***Niveles de Capacidad*** de acuerdo a parámetros necesarios para ir ascendiendo niveles. Para que esto se lleve a cabo, es necesaria una colaboración de toda la organización, es decir, desde participantes en proyectos, hasta administradores o gerentes de alto nivel que no intervengan en proyectos específicos.

MoProSoft está basado en procesos, y muchos de los procesos incluidos en él salen fuera del control de un proyecto específico (por ejemplo, del proyecto de desarrollo del Web LTST), es decir, son procesos que se llevan a cabo por altos niveles gerenciales de la organización o por personal de soporte que son independientes de la realización de proyectos

Por esta cuestión, no es imposible manejar aspectos que están fuera de nuestra competencia, como la administración de personal, estructura organizacional, recursos materiales, etc. Para trabajar con ***MoProSoft*** exclusivamente para el desarrollo del sitio, se hicieron las consideraciones necesarias para poder demostrar que se trabajó con este marco de procesos. Por tanto, el presente capítulo es en gran medida la explicación de cómo se llevó a cabo la construcción del sitio bajo su contexto.

El desarrollo del sitio siguió lineamientos del marco de procesos de ***MoProSoft***, por lo tanto esto conlleva que el trabajo se desarrolló dentro de un Nivel de Capacidad. En la sección 6.7, se presentará el Nivel que se ha logrado.

En los siguientes apartados, se mencionan aquellos documentos fundamentales y representativos que describen la forma en que se construyó el sitio, en la sección 6.1 Nivel de Capacidad se explican los parámetros y los niveles que alcanza una organización de acuerdo a los procesos implementados y que además, el trabajar bajo esta metodología, se puede estar sujeto a una evaluación. En el capítulo 6.2 Base del Conocimiento, se explica la estructura y la organización física de los productos de trabajo al implementar el marco de trabajo. En los apartados 6.1 a 6.6.2 se explicarán en qué consiste cada uno de los procesos, sus principales productos de trabajo y el rol que juega cada uno de ellos en las diferentes etapas del desarrollo del proyecto. Finalmente, como subtemas del apartado 6.6.2 Desarrollo y Mantenimiento de Software, se implementa la metodología del Proceso Unificado, la cual tiene que ver principalmente con el desarrollo del producto de software, por lo que el apartado se ha dividido en las principales fases de trabajo del Proceso Unificado.

1.1. Nivel de Capacidad

El desarrollo del sitio Web LTST tiene como uno de sus objetivos adoptar el marco de trabajo de ***MoProSoft.*** Cuando una organización trabaja bajo estos lineamientos, ésta puede adquirir un Nivel de Capacidad de acuerdo a parámetros y criterios que se hayan cumplido en el desarrollo de sus procesos. Para este proyecto, Web LTST, se ha puesto como uno de los objetivos cumplir con un Nivel de Capacidad 1, debido a aspectos importantes que se irán explicando en el transcurso de esta sección.

El sitio Web del LTST es un proyecto interno del mismo laboratorio; el trabajar en un proyecto interno de una organización cuyas actividades centrales no se dedican al computo, dificulta aplicar los lineamientos bajo los cuales hay que trabajar en los procesos del marco de trabajo de ***MoProSoft***, una gran desventaja es que algunos de los procesos como la ***Gestión de Recursos*** y los subprocesos ***Recursos Humanos y Ambiente de Trabajo***, ***Bienes, Servicios e Infraestructura*** y ***Conocimiento de la Organización,*** están fuera de nuestro control, pues no intervenimos en la toma de decisiones y manejo de los aspectos que involucran estos procesos, los cuales se enfocan hacia el contexto general de la organización y no es posible implementarlos dentro de un proyecto específico.

En el documento de ***MoProSoft***, se detallan todos los productos y tareas que deben realizarse, éstos son una serie de prácticas que atienden a cuestiones técnicas y administrativas en todos los procesos, incluyendo orden, control, administración, planeación, mantenimiento, optimización, etc. Y que son parámetros que están ligados a los ***Niveles de Capacidad*** que se otorga a las organizaciones.

EvalProSoft es el documento que evalúa y dictamina los parámetros necesarios para otorgar ***Niveles de Capacidad*** según los requerimientos de ***MoProSoft***. La Secretaría de Economía es la encargada de dicho modelo de Evaluación de Procesos, sin embargo aún no se ha publicado un documento que especifique a detalle los parámetros que definen los criterios de evaluación.

De acuerdo a Hanna Oktaba, directora del proyecto de Método de Evaluación de Software, EvalProSoft, los ***Niveles de Capacidad***, están sujetos a[1]:

- **Nivel 0 Incompleto**
  - No se cumplen los propósitos del proceso.
  - No se identifican fácilmente cuales son los productos del trabajo o salidas del proceso.
- **Nivel 1 Realizado**
  - Se alcanza el propósito del proceso. Aunque puede no estar rigurosamente planeado y rastreable.
  - Están identificados los productos del proceso que testifican que se alcanzó el propósito.
- **Nivel 2 Administrado**
  - El proceso realizado se implementa de manera administrada (planeado, supervisado y ajustado).
  - Los productos de trabajo están establecidos, controlados y mantenidos.
- **Nivel 3 Establecido**
  - El proceso administrado se implementa mediante el proceso definido.
  - Cada implementación individual del proceso sigue estándares aprobados, revisados y documentados.
- **Nivel 4 Predecible**
  - El proceso establecido opera dentro de ciertos límites para alcanzar sus resultados.

1 Presentación "EvalProSoft y Pruebas Controladas". Hanna Oktaba, UNAM, AMCIS.

- **Nivel 5 Optimización del proceso**
  - El proceso predecible se mejora continuamente para lograr las metas de negocio actuales y futuras.

Según estos niveles, los procesos seguidos en el desarrollo del proyecto se encuentran en Nivel 1, puesto que se tienen identificados los productos correspondientes a cada proceso, por lo tanto, en el desarrollo del capítulo se muestran solamente aquellos documentos que reflejen el proceso de construcción del sitio, pero se omiten algunos otros como reportes, minutas los cuales tienen por objetivo mantener cierto control o llevar un seguimiento de las actividades en sus diferentes fases, y lo cuál no es requerido en el Nivel 1.

Para llegar a un Nivel 2, considerando solamente aquellos procesos que están a nuestro alcance hay un requisito que por cuestiones tecnológicas no se podrá cumplir para este proyecto y es principalmente la cuestión de planeación. Este requisito es difícil de cumplir cuando se trabaja con tecnología relativamente nueva, es decir, incluir Flash no sólo enfocado a la animación, sino como parte de un sistema Web, el uso de Flex para el desarrollo de las RIA y éste incluirlo en un entorno J2EE; todo esto hace que se dificulte llevar a cabo una planeación y tener un proceso totalmente controlado, estos últimos son requerimientos indispensables para alcanzar un Nivel 2 en los procesos. Por estas razones se ha decidido desarrollar el proyecto dentro del Nivel 1 de Capacidad, atendiendo a requisitos mínimos e indispensables que en la sección 1.3 ***Gestión de Negocio*** se explican.

1.2. Base de Conocimiento

La ***Base de Conocimiento*** es un repositorio general donde físicamente se encuentran todos los documentos y archivos necesarios para el funcionamiento y desarrollo de toda la organización (incluyendo sus proyectos). De acuerdo al documento de ***MoProSoft***, en la versión 1.1 se manejan los siguientes repositorios:

- Negocio
- Procesos
- Proyectos
- Desarrollo y Mantenimiento
- Recursos
- Recursos Humanos y Ambiente de Trabajo
- Documentación BC (abarca documentos concernientes al manejo de ésta, como por ejemplo estructura, contenidos, documentación, etc.).

De acuerdo a ellos, se ha organizado la ***Base de Conocimiento*** LTST de la siguiente manera, bajo directorios de archivos Windows, tal como muestra la Figura 1.1:

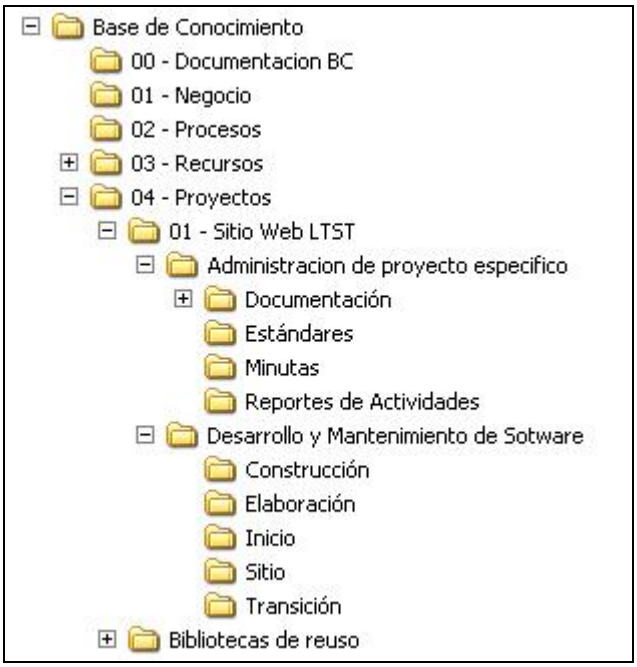


Figura 1.1 ***Base de Conocimiento*** **para el LTST**
Fuente: Elaboración propia.

Cualquier documento que se maneja dentro de la ***Base de Conocimiento*** tiene un campo obligatorio que marca el "estado" en que se encuentra el documento; existen tres estados:

- En desarrollo
- Verificar y validar
- Listo

La función de cada uno de los estados es indicar en qué etapa se encuentra un documento, ***en desarrollo*** nos indica un documento que aún no se encuentra terminado su contenido, ***verificar y validar*** indica que el documento ha sido terminado, sin embargo debe ser revisado y autorizado por una persona capacitada para dar la aprobación de que el documento es correcto, por lo mismo, se deberá anexar un campo con el nombre de la persona que revisó el documento, cuando el documento este terminado y correctamente revisado se pasará al estado ***listo***.

En el repositorio de ***Administración de Proyectos Específicos*** se encuentran todos los documentos no técnicos del proyecto incluye los repositorios concernientes a: documentación de tesis, estándares (plantillas y formatos), minutas y reporte de actividades. En el repositorio de ***Desarrollo y Mantenimiento de Software*** se encuentran todos los documentos técnicos y los archivos de construcción de sitio, incluyendo código fuente, archivos compilados, etc. A este repositorio se añaden los correspondientes a las fases de desarrollo del Proceso Unificado: Inicio, Elaboración, Construcción y Transición y en ellos están los documentos correspondientes a estas fases.

Es importante mencionar que ***MoProSoft*** no generaliza la forma de organización interna de la ***Base de Conocimiento*** y solamente indica cuales son los repositorios generales que deben manejarse.

1.3. Gestión de Negocio

Este proceso se encuentra dentro de la Alta Dirección, le compete principalmente al nivel más alto de una organización. La ***Gestión de Negocio*** tiene como propósito establecer elementos que caracterizan a la organización, como sus objetivos, su contexto, políticas de la empresa, su razón de ser, posibles mejoras, fortalezas, debilidades, etc. Uno de sus grandes propósitos es preparar a toda la organización para trabajar bajo lineamientos con el fin de alcanzar propósitos establecidos.

Trabajar con ***MoProSoft*** indica que el proyecto de software y la organización pueden estar sujetos a una evaluación. Como ya se mencionó, EvalProSoft es el documento que evalúa y dictamina los parámetros necesarios para otorgar ***Niveles de Capacidad*** según los requerimientos de ***MoProSoft***. Sin embargo, cuando se inició este proyecto no había un documento oficial que detallara los parámetros necesarios que definen cada uno de los Niveles de Capacidad, y al término del proyecto, aunque ya existe ese documento avalado por la Secretaría de Economía, pero que aún no es público su contenido, se optó por adoptar un estudio de análisis profundo y claro llevado a cabo dentro del Instituto de Ingeniería de la UNAM, ha determinado definir un modelo de evaluación propio para ***MoProSoft*** basado en el ISO/IEC 15504 y el documento de definición de ***MoProSoft*** en su versión 1.1.

El trabajo realizado cita como fin o propósito del documento: "presentar un Modelo de Evaluación de Procesos para ***MoProSoft*** (alterno al próximo a ser publicado por Secretaría de Economía) que forme la base para la recolección de evidencia y la evaluación de capacidades de procesos, con el fin de asegurar que los resultados de evaluación sean traducibles a un perfil de procesos ISO/IEC 15504 de una manera repetible y confiable." Por lo tanto, a través de este documento y cuyo título tentativo es "Modelo de Evaluación de Procesos de Software del II-UNAM", se han contemplado qué parámetros son suficientes para ser poder evaluar y asignar un ***Nivel de Capacidad*** a los procesos correspondientes en el desarrollo del sitio Web.

Como parte de este capítulo se muestran los documentos que se consideraron convenientes para comprobar que se realizó un desarrollo dentro de los lineamientos de ***MoProSoft,*** a pesar de que no exista por el momento un documento público oficial donde se especifique claramente aquellas tareas mínimas obligatorias dentro de un marco de evaluación. Los documentos indispensables se irán mencionando conforme al desarrollo del capítulo.

Dentro del contexto de la ***Gestión de Negocio***, uno de los principales documentos que caracteriza este proceso es el ***Plan Estratégico de la Organización*** y éste debe contener la siguiente información[2]:

- La Misión, Visión y Valores, que constituyen la política de calidad de la organización.
- Los Objetivos de la organización, incluyendo los objetivos de calidad, así como la forma de alcanzar éstos por medio de la definición de Estrategias.
- La forma de medir el logro de los Objetivos, por medio de la definición de Indicadores y Metas Cuantitativas asociadas a dichos Objetivos.
- Los Procesos Requeridos para alcanzar los Objetivos, con sus indicadores y metas.
- La Cartera de Proyectos que habilite la ejecución de las Estrategias.
- La Estructura Organizacional y Estrategia de Recursos que soporten la implantación de los procesos y la ejecución de los proyectos definidos, considerando los elementos de la ***Base de Conocimiento*** necesarios para el almacenamiento y consulta de la información generada en la organización.
- El Presupuesto, el cual incluye los gastos e ingresos esperados.

2 "Modelo de Procesos para la Industria de Software MoProSoft Versión 1.1 Mayo 2003"

- Periodicidad de valoración del plan estratégico.
- Plan de Comunicación con el Cliente, incluye los mecanismos de comunicación con el cliente para su atención.

Los puntos anteriores están considerados dentro del ***Plan Estratégico del LTST***, todos ellos enmarcan la razón de la organización y parte del contexto interno y externo que le rodea; por lo que se pretende que cualquier proyecto y todo el trabajo que se realice deben estar apegados a los puntos que se manejan en este documento.

1.4. Gestión de Procesos

En la ***Gestión de Procesos*** se establecen los procesos a seguir por la organización, de acuerdo a criterios y puntos establecidos en el documento ***Plan Estratégico del LTST***. El Documento de ***MoProSoft, versión 1.1,*** es una guía que indica los productos, tareas, roles, actividades y procesos que debe tener la organización, en este caso el LTST.

Para el desarrollo del proyecto no se han implementado la totalidad de los procesos ni se han creado el 100% de los documentos establecido para los procesos implementados, esto se debe a que desde el punto de vista del proyecto sólo se consideran una parte de los procesos implementados, y de ellos solamente el alcance del primer nivel de capacidad logrado, de acuerdo a lo especificado en el documento "***Modelo de Evaluación de Procesos de Software del II-UNAM***".

1.5. Gestión de Recursos

El proceso de ***Gestión de Recursos*** es quizá el proceso más difícil de cubrir si no se cuenta con la participación de toda la organización; esto se debe a que el principal objetivo de la ***Gestión de Recursos*** es obtener y proveer de infraestructura humana y de equipo, crear un ambiente de trabajo, mantener relaciones con proveedores, asignar roles a los integrantes, realización de evaluaciones, otorgar capacitación al personal, entre muchas otras tareas. Consecuentemente afecta a sus subprocesos de ***Conocimiento de la Organización***, ***Recursos Humanos y Ambiente de Trabajo*** y ***Bienes, Servicios e Infraestructura***.

Las tareas y actividades mencionadas son propias de personas fuera del proyecto del sitio Web, por lo que es uno de los procesos donde no se tiene una participación importante y por tanto no se puede asignar un ***Nivel de Capacidad***. La mayor parte de los productos que se piden no puede aterrizarse sobre proyectos en específico, sino que se contemplan de manera conjunta y global para toda la organización.

Sin embargo, uno de los puntos factibles de este nivel es la creación de la ***Base de Conocimiento***, la cual ya se explicó en apartados anteriores. Además de crearla y estructurarla, es necesario realizar algún plan de administración, de manejo y de mantenimiento. Otra forma en que se interactuó con este proceso es por medio de las solicitudes de recursos y los informes sobre utilización de los mismos.

1.6. Gestión de Proyectos

El proceso de ***Gestión de Proyectos*** tiene como finalidad el cumplimiento de los objetivos trazados para la organización por medio de proyectos internos y externos.

Algunas de las prácticas que se contemplan en esta parte es la captación de clientes, realización de estudios para cubrir las necesidades que el cliente pide y también lo relacionado con la evaluación de

proyectos para obtener estimaciones de costos y periodos de tiempo. Principalmente se establecen contratos, si así se amerita, o bien, se registra y se realiza un documento con una descripción general del proyecto; así mismo se asigna a un responsable para cada proyecto.

Los principales documentos a considerar son: ***Registro de Proyecto***, ***Contrato*** (si aplica), ***Descripción del Proyecto***, ***Metas Cuantitativas*** y se asigna un Responsable del Proyecto o producto. A continuación se detalla qué se contempló para el sitio Web dentro del documento de ***Descripción del Proyecto***, los tres documentos restantes, están considerados en la ***Base de Conocimiento***, sin embargo no se muestran, pues tecnológicamente no son relevantes para el desarrollo del sitio.

Descripción del proyecto

El proyecto consiste en crear y diseñar el sitio en Internet del Laboratorio de Transporte y Sistemas Territoriales, perteneciente al Instituto de Ingeniería de la UNAM. En los siguientes puntos se describe el objetivo, en qué consistirá el sitio, la necesidad del proyecto, restricciones, etc.

Objetivo de la propuesta:

Desarrollar un sitio en Internet para el Laboratorio de Transporte y Sistemas Territoriales del Instituto de Ingeniería de la UNAM. Para este sitio se adoptarán el Modelo de Procesos para la Industria de Software (MoProSoft) y prácticas internacionales de modelado (UML), y además se implementarán los lineamientos del Proceso Unificado de Desarrollo de Software.

Definición:

Se creará un sitio en Internet para que sea un vínculo de presencia internacional para el Laboratorio de Transporte y Sistemas Territoriales (LTST) en el mundo. Este proyecto tendrá como meta principal informar de las actividades, historia y personal que labora en él, así mismo se pretende que sea un espacio de promoción y captación de trabajos relacionados a la Ingeniería de Transporte y a la Logística.

El sitio será construido siguiendo una metodología basada en el Proceso Unificado de Desarrollo de Software, por lo que será un sitio apegado a las normas de calidad, además de adoptar el marco de trabajo MoProSoft. Además, el sitio quedará preparado para que puedan agregarse nuevas funcionalidades que no estén contempladas actualmente.

Metodología

Al implementar el modelo de procesos de MoProSoft se buscará hacer un producto que satisfaga los objetivos trazados para este marco de trabajo.

El desarrollo del sitio será implementado en tres etapas: Creación de la parte informativa, creación de la parte restringida a usuarios registrados y finalmente creación de la parte sólo permitida a usuarios del LTST. Estas 3 etapas estarán divididas en cuatro fases, las cuales son inicio, elaboración, construcción y transición, cada una de ellas divididas en una serie de pasos que se detallan a continuación.

Inicio

- Realizar una investigación sobre los requerimientos del cliente, contemplando cambios, nuevos requerimientos, según sea el caso de la fase en que se encuentre.
- Realizar un análisis inicial de los requerimientos encontrados.
- Estudiar las tecnologías disponibles para el desarrollo.

- Elaborar y probar ejemplos didácticos en las tecnologías disponibles.
- Se espera contar con una identificación del 80% de los casos de uso.

Elaboración

- Afinar y detallar los requerimientos del sitio con base en la fase anterior.
- Establecer el tipo de tecnología a seguir para el desarrollo de sitios en Internet.
- Con base en la investigación de los requerimientos, efectuar un estudio de factibilidad técnica y operacional. El primero consiste en estudiar si el trabajo para el proyecto puede desarrollarse con el software y hardware existentes, y en caso de necesitar nueva tecnología, identificar las posibilidades de adquirirla; el segundo estudio consiste en investigar quiénes serán los usuarios finales del sistema y qué capacidad tendrá éste para proporcionarles beneficios.
- Después de haber analizado todos los requisitos del proyecto así como los recursos disponibles, se estimará el tiempo necesario para la conclusión de la etapa.
- Se espera contar con una identificación del 95% de los casos de uso, además de tener el 80% del análisis de los casos de uso ya identificados.

Construcción

- Se comenzará a utilizar paquetería especializada para la creación de sitios, se realizará la programación necesaria para implementar funciones específicas del sitio y cumplir con los, requerimientos establecidos, además se alaborarán los documentos correspondientes a MoProSoft.
- Las pruebas de diseño e implementación se enfocan en la estructura interna de la unidad, confiando en la implementación de la unidad (pruebas de caja blanca). Para la verificación del comportamiento y función de la unidad, no importa la forma en que se haya implementado (pruebas de caja negra).
- Se espera tener el 98% de los casos de uso identificados, y desechar el 100% de los casos de uso no factibles.
- Se deberá tener analizado el 100% de los casos de uso identificados y el 85% de los casos de uso ya implementados en el sitio Web.

Transición

- Durante esta etapa se realizaran pruebas unitarias en el sitio, para asegurarse que el software no tenga fallas, es decir, que funcione de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo haga. Las pruebas unitarias se implementan para probar los elementos más pequeños del software, e involucra probar su estructura interna y el flujo de datos dentro de estos elementos, así como su comportamiento.
- Se espera tener implementado el 100% de los casos de uso identificados y tener un margen de tiempo considerado por algún requerimiento o cambio factible de realización, que no signifique algún cambio en los casos de uso ya implementados.
- Se realizarán las pruebas Funcionales del Sitio, éstas se basan en los casos de uso ya implementados.
- Se hará entrega del sitio de Internet, incluyendo documentación; además, se hará la instalación de la aplicación y se construirán todos los archivos de datos de desarrollo de aplicaciones subsecuentes.

Como resultado final se espera contar con un sitio eficiente y cuya funcionalidad sea exitosa en cuestiones tecnológicas, y de utilidad a los usuarios, tanto aquellos que pertenezcan al equipo de trabajo del Laboratorio de Transportes y Sistemas Territoriales, como a aquellas personas externas con las que se deseaba tener una interacción.

El documento anterior describe de una forma global las expectativas que se deben cumplir y también contempla las tareas generales que deben ser realizadas para cada flujo de trabajo, bajo la metodología de Proceso Unificado.

Metas Cuantitativas

Las metas cuantitativas ayudarán a definir los objetivos correspondientes a tiempos durante la construcción del sitio. Es recomendable hacer un seguimiento de estas metas a fin de evaluar si se han ido cumpliendo o no. Las metas cuantitativas establecidas son las siguientes:

- El proyecto del sitio Web se pretende terminar en un periodo máximo de 9 meses (enero 2005 – septiembre 2005).
- Se irán haciendo entregas de avances, sin fechas por definir, es decir, se irán pactando según el avance. Las entregas se distribuirán de la siguiente manera:

  Parte informativa y estática del sitio.
  Registro, Validación y parte permitida sólo a usuarios registrados (noticias y Publicaciones).
  Parte Exclusiva de miembros del LTST.

- Se pretenden tomar cursos de capacitación, uno en tecnología Flash y otro en tecnología .Net. Ambos durante los primeros tres meses del año, con la finalidad de aprender y desarrollar al mismo tiempo.
- Se dispondrán de 2 computadoras para implementar el sitio:

  PC con procesador Intel P4 a 2.8 GHz, con 512 MB de RAM y DD 80GB
  PC con procesador Intel P3 a 450 MHz, con 256 MB de RAM y DD 80GB

- Se realizará, en caso de ser necesaria, la adquisición de material bibliográfico; se hará la solicitud para su adquisición, que será sujeta a aprobación.

Con la elaboración de los documentos previamente citados se finaliza el proceso de ***Gestión de Proyectos***. A continuación se desarrollan los subprocesos de ***Administración de Proyectos Específicos*** y ***Desarrollo y Mantenimiento de Software***.

1.6.1. Administración de Proyectos Específicos

El proceso de ***Administración de Proyectos Específicos*** establece todo lo necesario para cumplir el proyecto administrada y sistemáticamente, de acuerdo a los tiempos y costos estimados. Esta parte de la ***Base de Conocimiento*** puede administrarse por proyectos o por conjunto de proyectos, y pueden ser establecidos repositorios que cumplan alguna función específica, como repositorios para contener estándares y formatos de los documentos, minutas y reportes de actividades (ver la Figura 1.2).

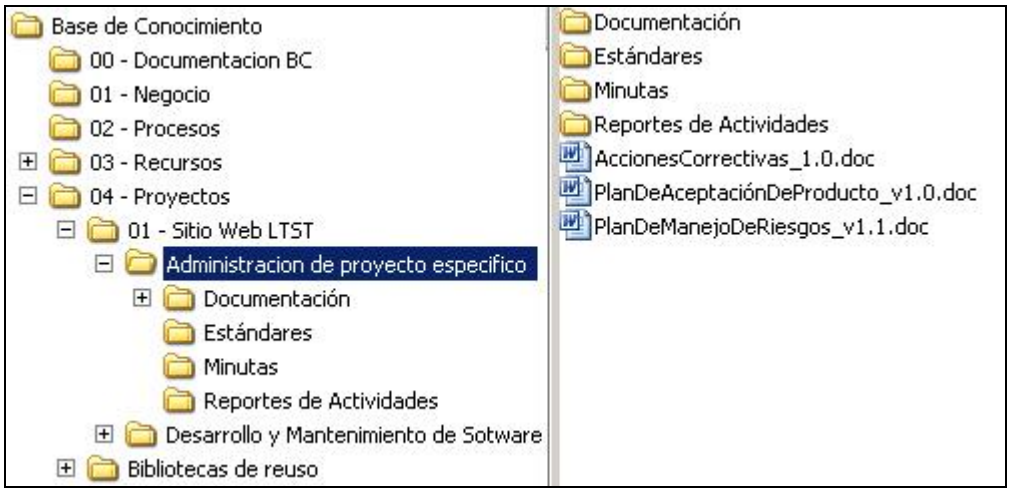


Figura 1.2 Organización de la *Administración de Proyectos Específicos*
Fuente: Elaboración propia.

Como en los demás procesos, hay documentos que son indispensables para cumplir con el marco de trabajo de ***MoProSoft*** en Nivel de Capacidad 1. Los documentos a considerar son: ***Plan de Manejo de Riesgos***, ***Reporte de Seguimiento***, ***Acciones Correctivas*** y ***Documento de Aceptación***. A continuación se muestra un resumen de los documentos ***Plan de Manejo de Riesgos*** y ***Acciones Correctivas***, con el objetivo de mostrar los riesgos más representativos al construir el sitio, pero también, mostrar las medidas que se tomarían para combatirlos. Los otros documentos no se muestran en este trabajo de tesis, aunque están elaborados dentro de la ***Base de Conocimiento***.

Plan de Manejo de Riesgos (Resumen)

El Plan de Manejo de Riesgos describe los principales y posibles riesgos que se pueden tener en el proyecto del sitio del LTST. Se han identificado los Riesgos más representativos y de mayor peso que se pudieran presentar en la construcción del sitio. Aquellos riesgos no considerados pueden ser anexados y tomados en cuenta dentro de este Plan de Manejo de Riesgos.

En la siguiente tabla, se describe la severidad y la probabilidad de ocurrencia del riesgo, de acuerdo a aquellos que se han identificado como los más representativos para el desarrollo del sitio.

La probabilidad de ocurrencia va de 1 a 5, siendo el 5 un nivel muy alto, 4 nivel alto, 3 nivel medio, 2 nivel bajo y 1 nivel muy bajo.

La severidad es qué tanto pueda impactar para el desarrollo del proyecto. 1 significa que se trata de un impacto pequeño, y cuya solución no es difícil de resolver, mientras que el 5 representa la escala máxima y es cuando existen mayores dificultades e imprevistos para continuar con un desarrollo constante del proyecto.

IR	Descripción de Riesgo	Probabilidad de Ocurrencia	Severidad
R1	Falta de conocimiento de las tecnologías que se están empleando, por ejemplo Macromedia Flex, ActionScript y Struts.	5	5
R2	La interfaz del sitio por ser desarrollada en flash, requiere de un reproductor de flash instalado en la máquina del usuario	2	1
R3	Dependencia de la velocidad de conexión por parte del usuario, que puede ir desde conexión bajo módem, con velocidades de 54kbps, a conexiones por banda ancha.	4	2
R4	Falta de información y/o publicaciones para completar bien las funcionalidades muy específicas del proyecto.	3	2

Principales riesgos enfrentados al desarrollar el sitio Web LTST

IR – Identificador de Riesgo

El documento de ***Acciones Correctivas*** no debe confundirse con un Plan de Contingencia. Las Acciones Correctivas identifican una serie de pasos o actividades a seguir con el fin de restablecer un flujo normal de trabajo cuando algun problema se sale de control o sucede de forma inesperada. En un Plan de Contingencia, se consideran otras cuestiones tales como: evaluación de aquellos posibles daños internos y externos en una organización, tiempo de respuesta a esas eventualidades, prioridades de los casos, asignación de responsabilidades, etc. A continuación se muestra el documento de ***Acciones Correctivas***, correspondientes a los riesgos identificados en el documento de ***Plan de Manejo de Riesgos***.

Acciones Correctivas (Resumen)

El documento de Acciones Correctivas para el proyecto del sitio Web, está orientado al cómo se responderá a aquellos riesgos contemplados en el Documento de Plan de Manejo de Riesgos, es decir, se informa sobre las acciones y medidas empleadas para combatir los riesgos. Este documento muestra las acciones correctivas de los riesgos identificados en el documento *Plan de Manejo de Riesgos*.

IR – Identificador de Riesgo.
AC – Acción Correctiva.

IR: R1 AC
Se pospondrá el tiempo planeado para la conclusión del proyecto, además de alargar los periodos de documentación de la tesis, para solventar el conflicto. Se tiene pensado contactar personas, foros y adquisición de libros, además de consultar documentos electrónicos para enfrentar los problemas correspondientes a este punto. Se considerará tomar cursos para cubrir las necesidades de aprendizaje de las tecnologías a emplear.

IR: R2 AC
Cuando el usuario visita sitios en donde sea necesario el reproductor de Flash, generalmente el sistema operativo del usuario manda un mensaje pidiendo el Reproductor de Flash de Macromedia, éste se descarga desde al sitio en Internet de Macromedia, bajo previa autorización del usuario. De acuerdo a Macromedia, este reproductor (Flash Player) está instalado en el 98% de computadoras, lo que significa que son pocas las posibilidades de que algún usuario no pueda ver el contenido del sitio. Hasta cierto punto, no es un problema que pueda ser resuelto o cuya solución esté en manos de los desarrolladores, sin embargo se tomará en cuenta..
IR: R3 AC
Se reducirá el tamaño de los archivos swf. De antemano se sabe que desarrollar un sitio Flash, genera archivos pesados, sin embargo se ha decidido seccionar los archivos de tal forma que los usuarios sólo descarguen el archivo que deseen visitar. Es una ventaja que el archivo swf quede almacenado en la memoria de la computadora, por lo que si los usuarios lo conservan, esta descarga sólo se hace una vez. Se harán pruebas con diferentes velocidades en las tazas de transmisión y se buscará que caigan dentro de rangos muy aceptables, comparados con otros sitios construidos con tecnología similar. Si no se llegara a estar de acuerdo con la velocidad de respuesta del sitio, posiblemente se tendría que modificar la interfaz del usuario bajando la calidad de las imágenes y reduciendo la implementación de éstas en los archivos.
IR: R4 AC
En algunas ocasiones se retrasará la entrega de material para completar el sitio. Conforme se desarrolle el proyecto se ha adoptado la metodología de hacer una estructura o simular la información, para que después cuando ésta se tenga, simplemente se vacíe en los archivos correspondientes.

Hasta aquí, se han mostrado los documentos pertenecientes a la parte administrativa del proyecto. A continuación se detalla la parte técnica del proyecto.

1.6.2. Desarrollo y Mantenimiento de Software

El proceso de ***Desarrollo y Mantenimiento de Software*** se centra en los documentos técnicos. En este proceso se realizan actividades correspondientes al análisis, diseño, construcción, integración y pruebas de productos. Es de esperarse que existan varias versiones y revisiones de los documentos por los cambios y avances constantes.

El repositorio en la ***Base de Conocimiento*** concerniente a ***Desarrollo y Mantenimiento de Software*** se muestra en la Figura 1.3. Esta Base contiene las versiones y avances de archivos del sitio, y los repositorios concernientes a los flujos de trabajo del Proceso Unificado, cada uno con sus documentos correspondientes. Éstos se irán mostrando en el transcurso de este capítulo.

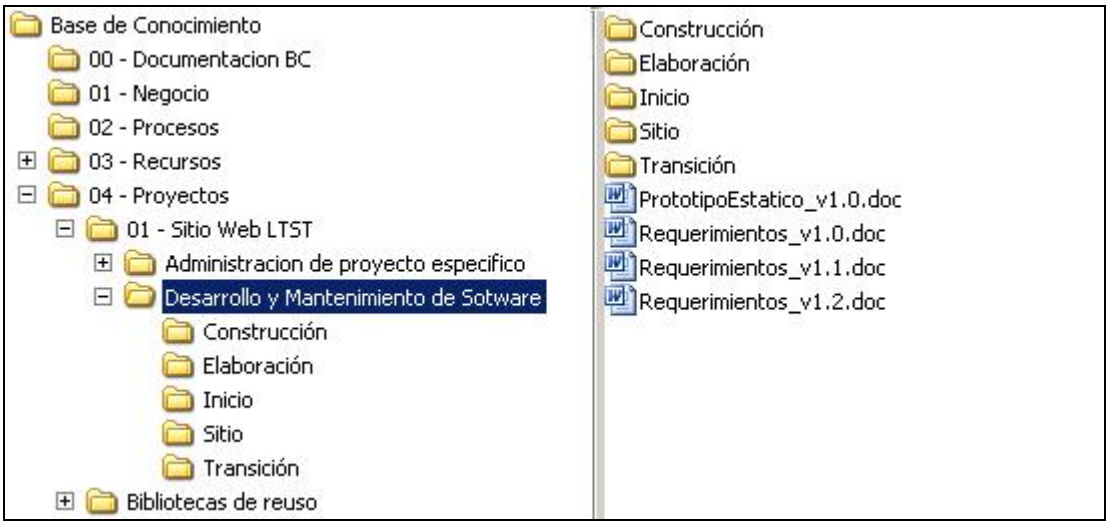


Figura 1.3 Organización de *Desarrollo y Mantenimiento de Software*
Fuente: Elaboración propia.

Los documentos mínimos para trabajar en el Nivel 1 de Capacidad son: ***Análisis de Requerimientos***, ***Análisis y Diseño***, ***Componente(s) de software***, ***Sistema Software***, ***Manual de Usuario***, ***Manual de Operación*** y ***Manual de Mantenimiento***. Estos documentos están distribuidos en las fases de trabajo del Proceso Unificado. En el ANEXO B se muestran las actividades llevadas a cabo durante el Proceso Unificado.

1.6.2.1. Requerimientos

El documento de ***Análisis de Requerimientos*** contiene la descripción del uso y funcionalidad del software basado en algunos de los siguientes aspectos: funcionalidad, interfaces (usuario, software y hardware), confiabilidad, eficiencia, mantenimiento, portabilidad, restricciones, etc.

Para el proyecto Web LTST, se establecieron los requerimientos generales en reuniones periódicas, en donde se mostraban avances, se pactaban cambios factibles y se establecían los objetivos a cumplir para la siguiente reunión.

Con base en los requisitos generales, establecidos en la primera reunión, se elaboró el documento de ***Análisis de Requerimientos***, el cual tuvo varias versiones a lo largo del periodo de construcción del sitio. A continuación se muestra el documento final, el cual contiene todos los aspectos que se consideraron:

Requisitos para el Sitio Web LTST

Los requerimientos funcionales del sitio Web LTST son los siguientes:

Parte Pública

- Información del Laboratorio

  En este punto se pretende dar a conocer la información más relevante del Laboratorio de Transporte y Sistemas Territoriales; aspectos como historia, filosofía (misión, visión, valores), historia, líneas o áreas de investigación y ubicación del LTST.

- Información del personal

  En esta parte se dará a conocer la información del personal que labora en el Laboratorio. El personal estará clasificado de acuerdo al nivel que tiene, el cual va desde becarios hasta investigadores. Se pretende estandarizar la información individual, y realizar una sola interfaz que unifique el contenido de la información.

- Proyectos

  Se mostrarán los proyectos realizados por el laboratorio, divididos en actuales y recientes. Se creará una lista con el nombre de los proyectos y al igual que en información del personal, se tendrá información particular de cada uno de ellos.

- Sitios relacionados

  Se trata de una lista de sitios relacionados al transporte, la cual incluirá el nombre de las diferentes instituciones. Además, al dar clic en el nombre, será desplegado el sitio correspondiente en una nueva ventana.

- Registro

  Se tendrá un formulario en donde el usuario deberá registrarse para obtener un nombre de usuario y contraseña para poder acceder a zonas sólo permitidas a usuarios registrados. El texto que haya introducido en el campo Nombre al igual que el correo electrónico escrito en el campo E-Mail corresponderá al login y password en la validación del usuario.

  Los campos obligatorios del registro son: nombre, correo electrónico, teléfono y nacionalidad y sexo. Los campos no obligatorios son: empleado, empresa, puesto, giro, ubicación; estudiante, institución y grado máximo de estudios.

Parte sólo permitida a usuarios registrados.

- Publicaciones

  Se pretende ofrecer información a Usuarios Registrados, tal como artículos, documentos, imágenes, etc.

- Noticias

  Es un servicio RSS (Really Simple Syndication) el cual esta implementado por medio de archivos XML y es utilizado específicamente por todo tipo de sitios que actualicen con frecuencia su información y desean distribuirla, se buscará servicios cuyas notas sean relacionadas con el transporte, o bien de información relevante. Aunque se tengan todo un apartado de noticias en general, también se pretende implementar un filtro donde se publiquen sólo las noticias relacionadas

con el transporte y logística. Si este filtro se implementa como parte de esta tesis, también será necesario crear mecanismos para almacenar las noticias en un periodo de tiempo y manejar un historial de ellas.

Parte privada del personal del LTST

- Diseñar un pizarrón virtual, es decir un lugar en donde puedan ser dejados avisos, mensajes o tareas, con funcionamiento similar al de Post – it software o al sitio:

  http://www.oscartrelles.com/projects/social/postit/

Requisitos No Funcionales

El sitio LTST debe:

- Ser fácil de utilizar.
- Permitir una fácil integración de nuevas funcionalidades.
- Ser desarrollado en plataforma J2EE.
- Ser accesible por red.
- Ser atractivo en cuanto al diseño.
- El sitio LTST no puede permitir el acceso directo por URL's, a áreas exclusivas a usuarios registrados.

Casos de Uso

Los Casos de Uso tienen por objetivo establecer los requerimientos establecidos para el sistema, describiendo qué se hace, más no cómo se hace. La siguiente figura muestra el diagrama general de Casos de Uso del sitio Web. Del caso de uso 1, Entrar Sitio LTST, se desprenden los casos de uso 1.1 al 1.8, los cuales representan las opciones del menú principal del sitio; de algunos de ellos se desprenden más casos de uso. En secciones posteriores se analizan y detallan mediante diagramas los casos de uso 1.2 Personal y 1.7 Registro.

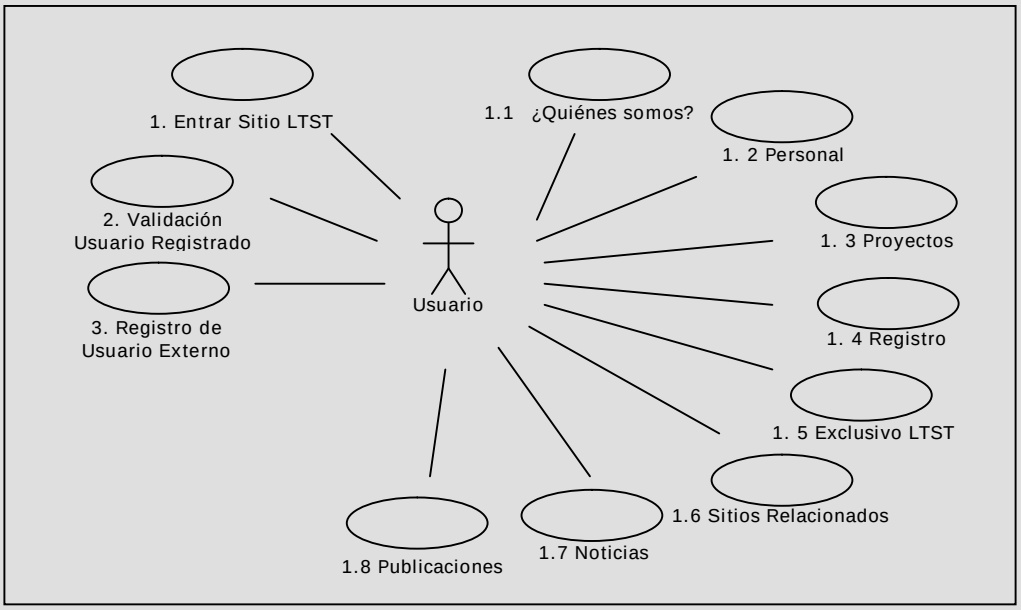
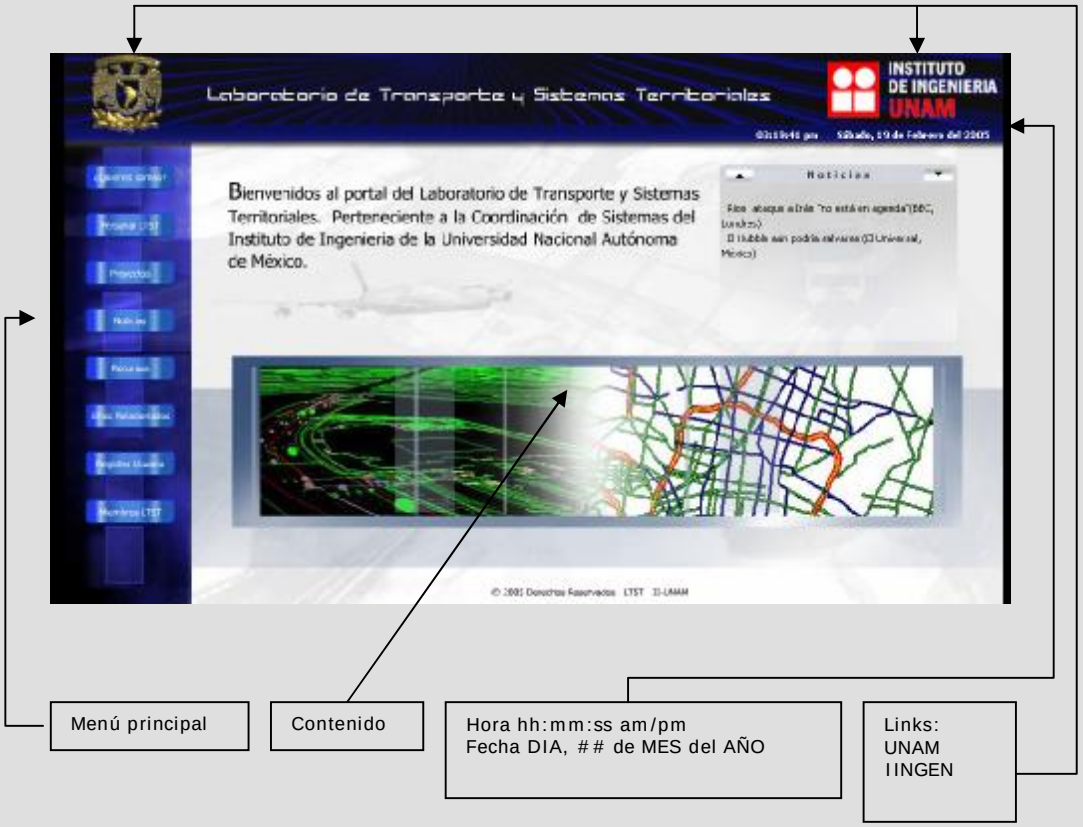


Figura Diagrama general de Casos de Uso del sitio Web LTST

Prototipo de Pantallas

La siguiente pantalla muestra el formato que tendrán todas las pantallas del sitio; solamente variará el contenido de ellas:



Se busca tener una interfaz que concuerde o no rompa con los estándares de colores y diseño del portal en Internet del Instituto de Ingeniería. La interfaz principal general del sitio contiene las siguientes secciones: encabezado, menú y área de contenido.

- El encabezado cuenta con los elementos: Escudos de la UNAM e IINGEN, los cuales funcionan como enlaces a sus sitios correspondientes; Hora y Fecha, y animación LTST. El encabezado siempre permanecerá visible y podrá contener submenús horizontales si así fuera necesario.

- El menú principal también permanecerá siempre visible en las diferentes secciones de la estructura del sitio. El estado de los botones del menú cambia a un tono oscuro dependiendo de la sección donde se encuentre, para indicar en qué posición del sitio se encuentra el usuario.

- Contenido: se compone de cualquier tipo de contenido dinámico o estático.

Se pretende que la pantalla principal (mostrada arriba) sea de un alto impacto visual para los visitantes, y que reflejn las actividades o perfil de LTST.

Al ingresar a la URL http://www.iingen.unam.mx/ltst, se accederá a una pantalla de introducción que es una animación en la que se pretende mostrar las áreas de actividades, de investigación y de desarrollo que se realizan en torno al transporte de carga y logística. Terminando la animación o en cualquier parte de ésta, puede ser accedida la pantalla principal del sitio.

1.6.2.2. Análisis y Diseño

La fase de análisis y diseño consiste en mostrar gráfica y textualmente aquellas partes que constituyen el sistema, de una forma general. Se implementa la arquitectura MVC como modelo de desarrollo.

Arquitectura del Sitio Web LTST

Para comprender el funcionamiento y estructura de un sistema, éste debe verse desde diferentes puntos de vista. Estas vistas se basan en el rol que juegan las personas y su perspectiva al interactuar con el sistema. Los roles pueden ser de usuarios finales, programadores, diseñadores, analistas, líderes de proyecto, etc. La arquitectura de un sistema se define como el "conjunto de decisiones significativas acerca de la organización de un sistema de software, la selección de los elementos estructurales (y sus interfaces) de los que se compone el sistema junto con su comportamiento como tal y como se especifica en las colaboraciones entre estos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas cada vez mayores y el estilo arquitectónico que orienta esta organización."[3]

De acuerdo a la definición, la arquitectura de un sistema se basa en aspectos estructurales y de comportamientos, a los que se denominan Vistas, las cuales son una perspectiva particular de ver sistema. La Figura 1.4 muestra el diagrama general de la arquitectura de un sistema, las vistas de las que se compone cada parte de ella, y los diagramas empleados para la construcción del Sitio. La figura está basada en el esquema que maneja James Rumbaugh, en su libro "El Lenguaje Unificado de Modelado", dentro del capítulo "Sistemas y Modelos".

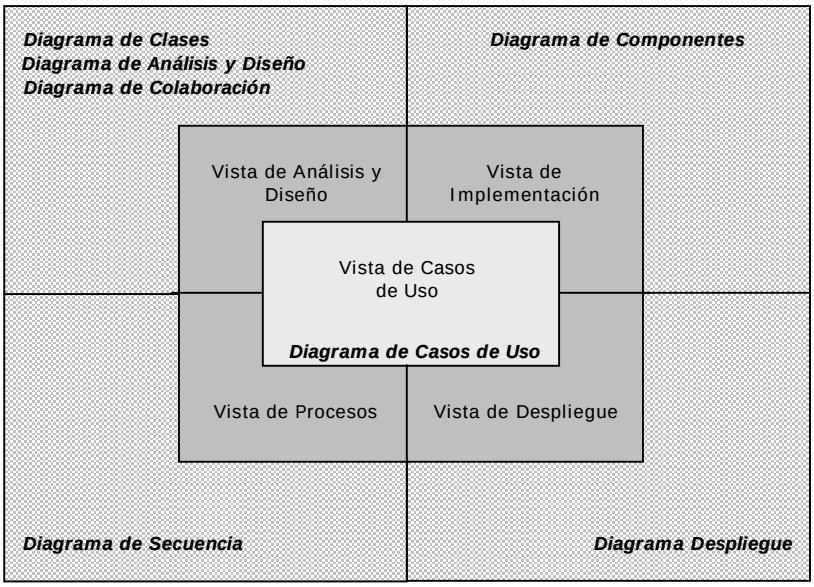


Figura 1.4 Arquitectura de un Sistema
Fuente: Elaboración propia.

[3] Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Lenguaje Unificado de Modelado"; Adisson Wesley Iberoamericana, Madrid 1999

Diagrama de Clases

El D***iagrama de Clases*** modela la estructura del sistema desde el punto de vista estático. El diagrama de clases se creó desde dos perspectivas, la primera, es una vista general y la segunda, se obtiene descomponiendo la vista general para obtener una particular. El ***Diagrama General de Clases*** está representado por medio de tres capas (Vista, Controlador y Modelo), las cuales se muestran como paquetes, cada uno de ellos conteniendo clases estereotipadas, las cuales constituyen la estructura del sitio. La Figura 1.5 muestra el ***Diagrama General de Clases*** del sitio.

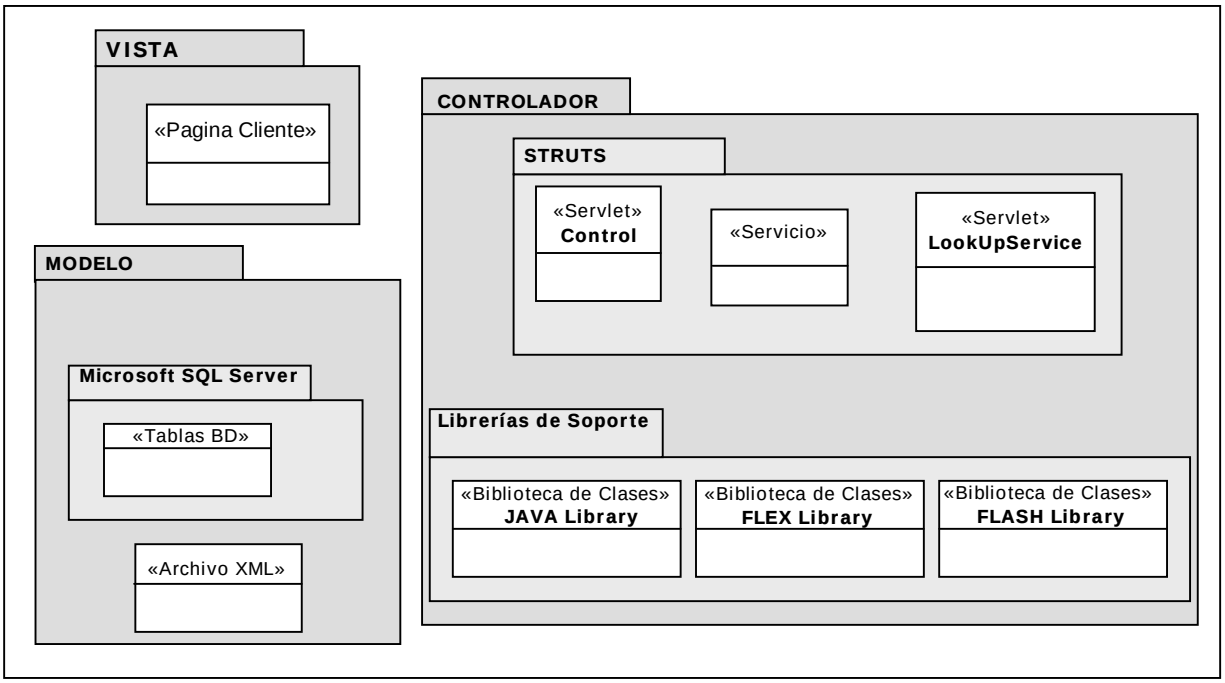


Figura 1.5 Diagrama General de Clases
Fuente: Elaboración propia.

En las siguientes figuras Figura 1.6, Figura 1.7 y Figura 1.8, se muestra el desglose de todas las clases que se derivan de cada una de las clases estereotipadas vistas en los paquetes de la figura anterior.

En la Figura 1.6 se observa que la clase Página Cliente representa todas las páginas que podrá manipular el usuario, la mayoría de ellas se derivan del ***Diagrama General de Casos de Uso***. La clase Integrante no existe como tal; Integrante es un archivo genérico que deberá existir por cada miembro del LTST y contendrá información personal, como por ejemplo su curricula y datos personales. Por cuestiones de distribución de espacios en la figura, no se muestran explícitamente las clases correspondientes a todos los integrantes, pues son aproximadamente 25 o 30 elementos, además de que estos archivos cambian frecuentemente, ya sea por alta o baja de personal o por actualizaciones frecuentes.

La capa Controlador se muestra en la Figura 1.7, contiene la aplicación del patrón ***Front Controller***, el cual provee un ***Control*** que maneja peticiones, interviene también la clase ***LookUpService*** que permite redireccionar las peticiones del cliente para ser procesadas y presentar una respuesta adecuada para esas peticiones. La clase ***Servicio*** contiene scripts necesarios para implementar la funcionalidad y procesamiento de peticiones por parte del cliente. Las ***Librerías de Soporte*** contienen todas aquellas librerías que pudieran ser usadas para la implementación del sistema.

Finalmente en el último paquete correspondiente al Modelo, Figura 1.8, están representados los datos contenidos en las bases de datos o archivos XML. Las clases de tipo XML Proyecto e Integrante, son la representación de una clase XML por cada miembros del LTST, y de igual forma la clase de tipo XML Proyecto es la representación para cada proyecto realizado en el LTST.

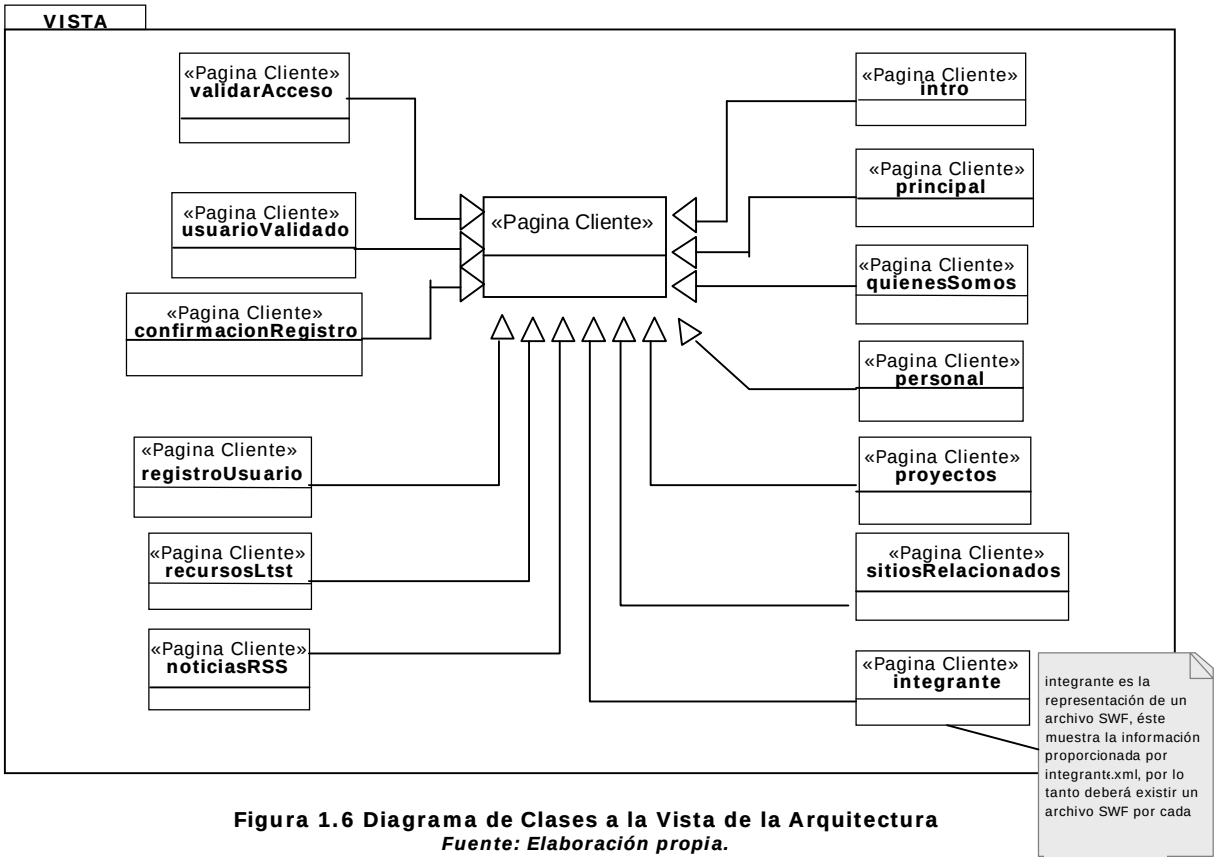


Figura 1.6 Diagrama de Clases a la Vista de la Arquitectura
Fuente: Elaboración propia.

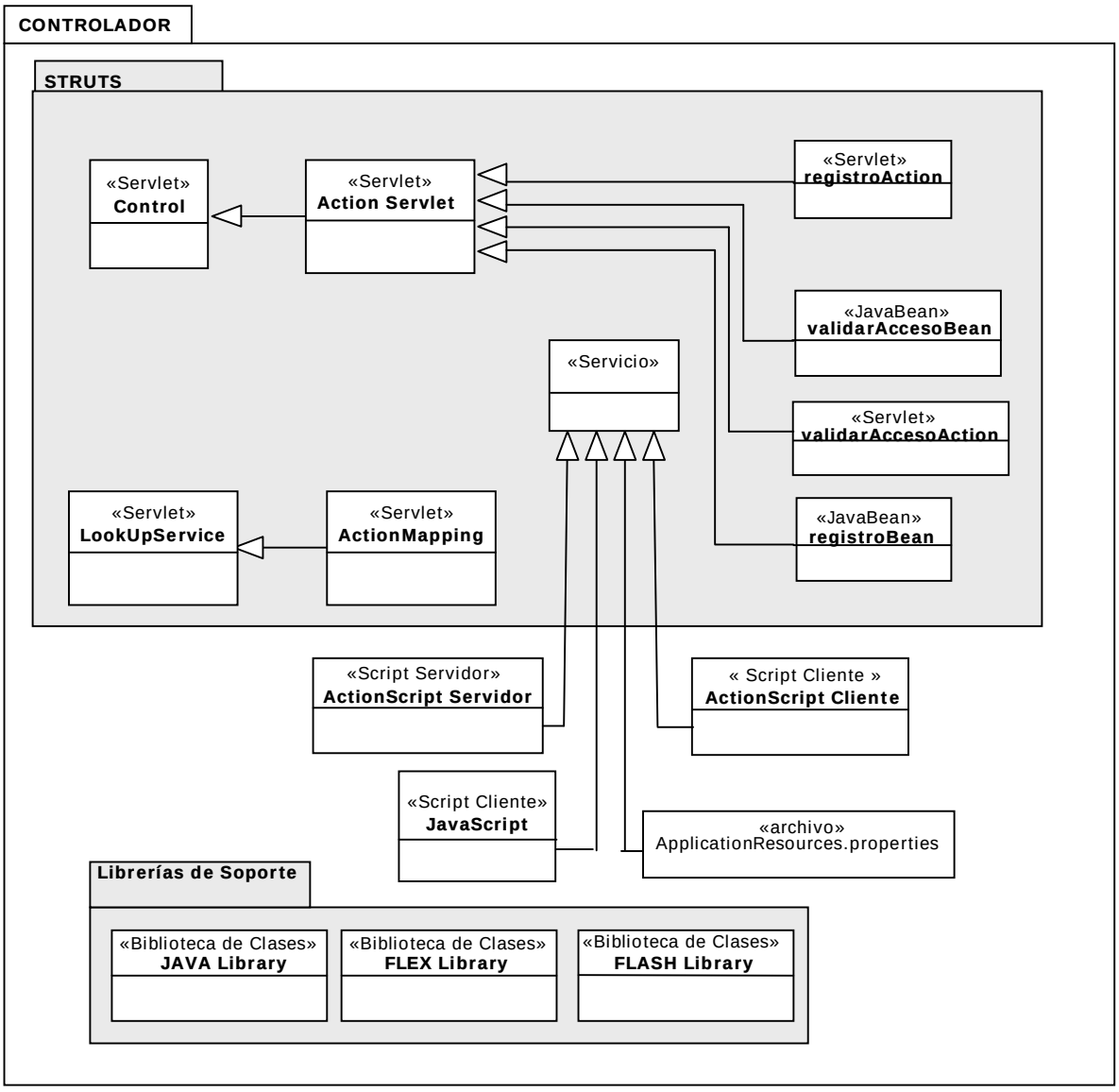


Figura 1.7 Diagrama de Clases correspondiente al Controlador de la Arquitectura
Fuente: Elaboración propia.

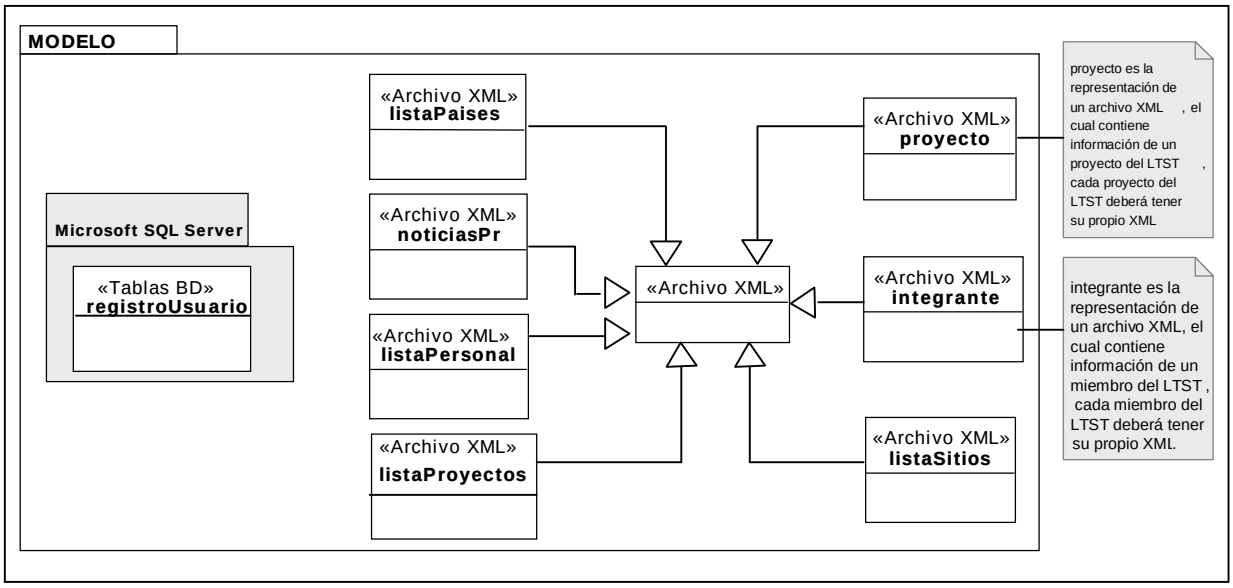


Figura 1.8 Diagrama de Clases correspondiente al Modelo de la Arquitectura
Fuente: Elaboración propia.

Los diagramas anteriores describen las clases a implementar en la estructura del sitio. No se han contemplado aquellas que estarán en la parte privada del sitio, debido a que por el momento se considera la opción de un pizarrón virtual, sin embargo, el desarrollo de éste puede afectar los tiempos estimados y objetivos principales del sitio. La implementación de esta parte se considera como un trabajo a mediano plazo.

Como se habrá observado, las clases aún no tienen definida la tecnología con la que se van a implementar, pero se espera sean clases de tipo Java, flash, xml y mxml las que estén involucradas.

1.6.2.3. Implementación

Diagrama de Componentes

El ***Diagrama de Componentes*** está conformado por el conjunto de los componentes del sistema y las dependencias, asociaciones o generalizaciones existentes entre ellos. En toda la estructura del sitio, estos diagramas están relacionados con los diagramas de clase, por lo que el diagrama de componente forma parte de la estructura estática de la arquitectura. Como parte de los diagramas de este tipo, se observan bibliotecas, tablas, archivos y documentos y relaciones entre ellos.

La Figura 1.9 muestra las tecnologías empleadas en la construcción del sitio. Agrupados en paquetes de forma vertical, se tienen las tecnologías, Java, Flex y Flash, cada una de éstas con clases propias de tecnología, por ejemplo en Java se observa la aplicación del patrón Front Controller. De forma horizontal, se mantienen los 3 paquetes pertenecientes a las capas Vista, Controlador y Modelo, que dividen las clases según donde cumplan su función.

Hay que recordar que Flex, implementa su propio patrón MVC, sin embargo, se trató de poner en un contexto coherente a la estructura del sistema.

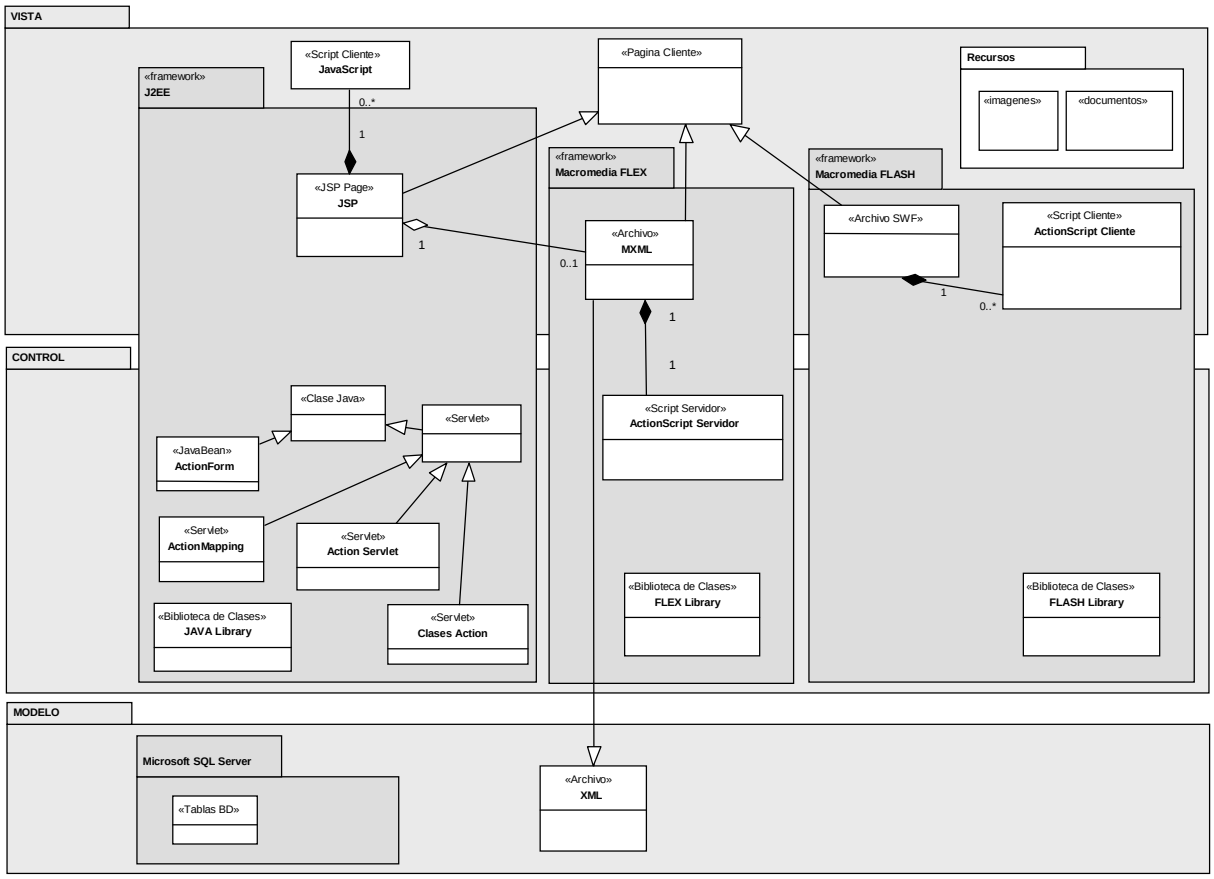


Figura 1.9 Diagrama general de Componentes
Fuente: Elaboración propia.

La Figura 1.10 muestra el ***Diagrama Particular de Componentes***, el cual descompone el ***Diagrama General de Componentes*** en clases específicas según la tecnología a la que pertenecen.

Diagrama de Despliegue

El último diagrama que representa a una de las vistas en su parte estática es el Diagrama de Despliegue. Sirve para modelar la relación entre hardware y software, y cubrir aspectos relacionados con el equipo sobre el que se ejecutará el software en forma eficiente. La

Figura **1.11 Diagrama de Despliegue**muestra el Diagrama de Despliegue del Sitio Web. El nodo cliente contiene el navegador Web del usuario. La única condición para poder visualizar el sitio, es tener el plug-in de Flash Player de Macromedia, debido a que son archivos swf.

El segundo nodo es aquel que contiene los servicios de aplicación del sitio (Flex y J2EE), todo está bajo la organización de archivos del servidor Tomcat 5.0, que es donde se ejecutan las aplicaciones Flex y J2EE. Para la organización del sitio se ha decidido colocar los archivos correspondientes a la parte de Flash, en el IIS de Windows, y la parte que requiere soporte J2EE en la estructura del servicio de Tomcat.

También se encuentra la base de datos registroUsuario, la cual almacena los datos enviados de la sección de registro. Como nota se especifican los requisitos mínimos que debiera tener el servidor de estas aplicaciones, debido a que Tomcat consume muchos recursos de memoria.

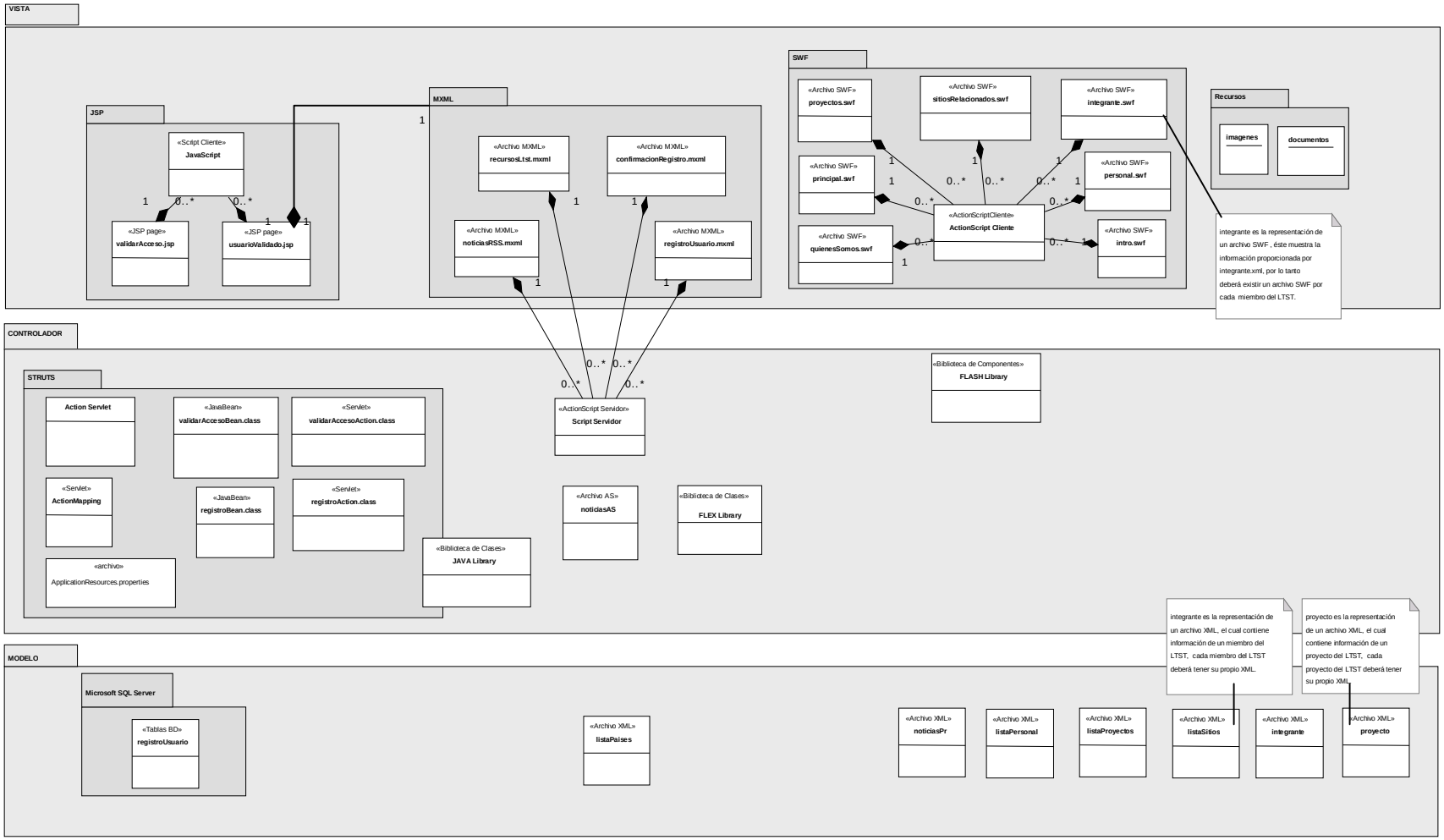


Figura 1.10 Diagrama Particular de Componentes
Fuente: Elaboración propia.

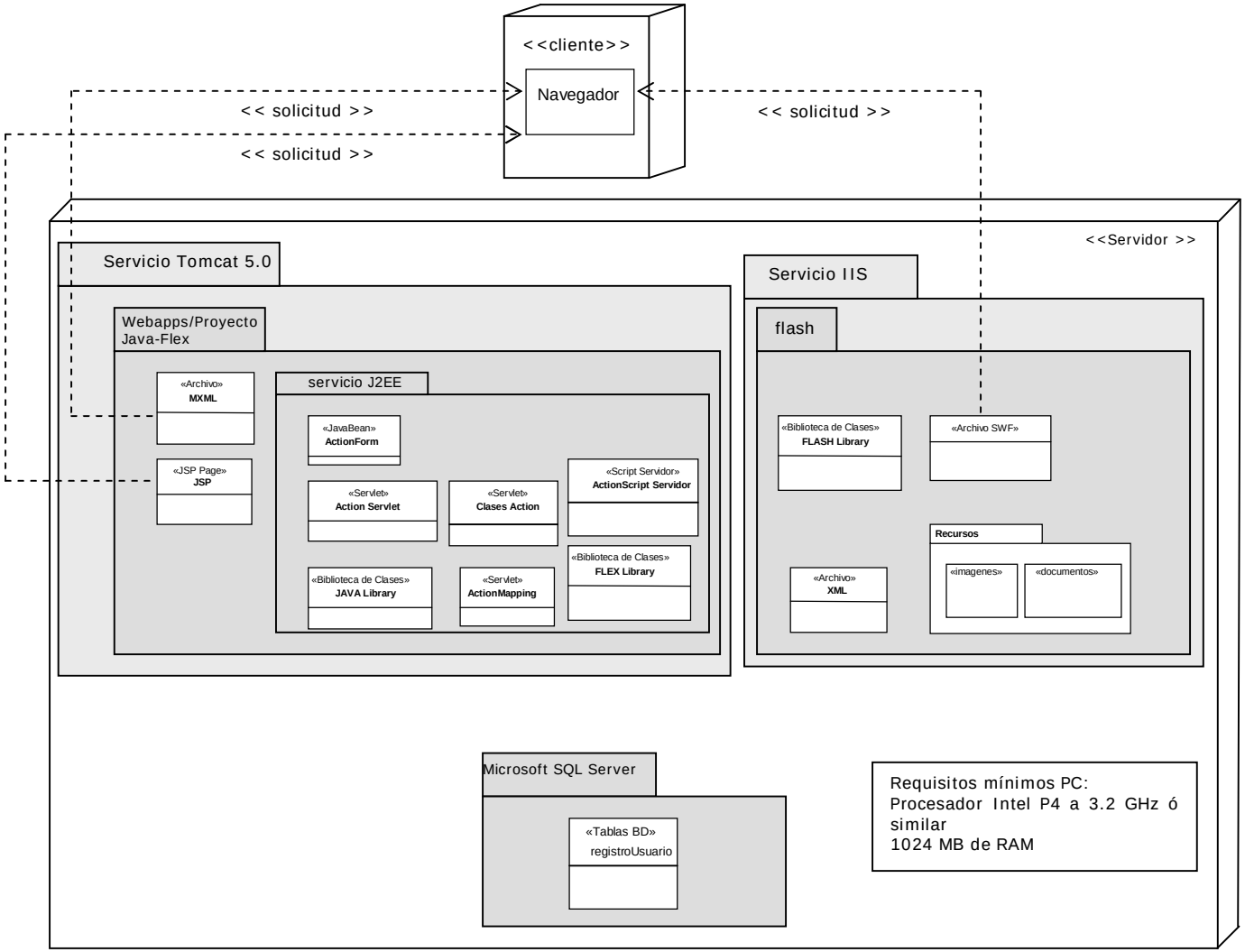


Figura 1.11 Diagrama de Despliegue
Fuente: Elaboración propia.

Hasta este momento se han cubierto los documentos de ***Análisis de Requerimientos***, los diagramas correspondientes al Análisis y Diseño y a la fase de Implementación. ***MoProSoft*** considera tres documentos más, Manual de Usuario, Manual de Operación y Manual de Mantenimiento. A continuación se muestran éstos.

Manual de Usuario

El manual de Usuario es poco aplicable para un sitio Web, por lo tanto se muestra el mapa de navegación del sitio.

Mapa de navegación

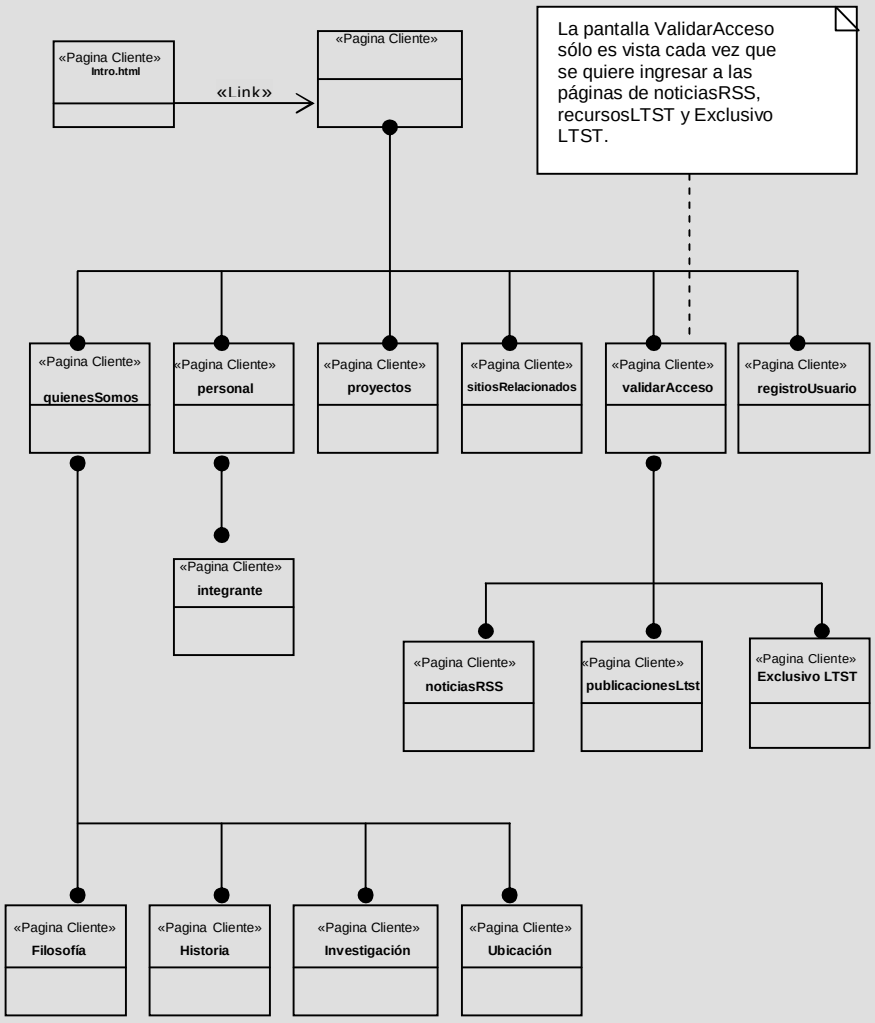
El mapa de navegación del sitio LTST, se basa en un menú principal, el cual aparecerá en todas las pantallas desplegables del sitio, a excepción de los casos Noticias, Publicaciones y Exclusivo LTST, en los cuales cambiará el menú.

Para representar el mapa de navegación por medio de UML, teniendo en cuenta que el contenido de las páginas html son Animaciones o Clips de Películas, es decir archivos de tipo swf, y en donde parte de la animación forma al Menú Principal del sitio; se ha decidido usar el término pantalla, en sustitución del término página, para no confundir con un posible modelado ya existente UML para Web. Al tener esta característica en la conformación del sitio Web, se ha creado una extensión UML para ésta:

Extensión UML para la aplicación Web: Link Múltiple

Nombre	Link Múltiple
Clase del Metamodelo	Múltiple asociación
Descripción	Este link permitirá asociar una pantalla con otra u otras, de tal manera que éstas también se podrán asociar con aquellas pantallas de las cuales hayan sido llamadas o llamar a otra paginas con las cuales no hayan tenido relación previamente.
Representación	

La siguiente figura muestra el mapa de navegación del sitio Web, cuyo funcionamiento es el siguiente. Se tiene una pantalla inicial que es intro.html, ésta está enlazada a la pantalla *Principal* por medio de un link; una vez que se accede a la pantalla principal se puede navegar a las otras pantallas por medio de un link múltiple, así por ejemplo se podrá llegar de la pantalla *Principal* a la pantalla de Quiénes Somos, y desde la pantalla de *Quiénes Somos* se podrá acceder a la pantalla *Principal* o a cualquier otra pantalla, por medio de un link múltiple.



De la figura, cabe aclarar que la representación en forma de clases *Filosofía*, *Historia*, *Investigación* y *Ubicación*, forman parte de un submenú al estar dentro del caso *Quienes Somos*. Nos son clases por sí mismas, toda la información que contienen está dentro del archivo *Quienes Somos*. Es por eso que desde el inicio de *Desarrollo y Mantenimiento de Software* nunca fueron incluidas como clases individuales, solamente hasta este momento y sólo para representar la navegabilidad del sitio Web.

Manual de Operación

El manual de operación contiene información relacionada con la instalación del software, y por ende del software que requiere para su funcionamiento.

7. RESULTADOS Y CONCLUSIONES

Antes, durante y después del desarrollo del sitio se detectaron ciertos factores, los cuales pueden llevar al éxito o fracaso un producto de software.

Uno de los primeros factores que deben ser tenidos en cuenta, es fijarse objetivos muy claros, por lo cual al inicio del desarrollo del proyecto se plantearon las preguntas: ¿cómo se va a hacer?, ¿con qué tecnología se va a hacer?, ¿cuándo se va a hacer? y ¿qué se necesita para desarrollarlo? La respuesta a todas estas preguntas se asentó en una idea general: se debe construir un sitio de Internet bajo un marco de trabajo y una metodología, proporcionados por la ingeniería de software; además se debe buscar un desarrollo con herramientas vanguardistas, fijarse un límite de tiempo para terminar el proyecto y buscar las mejores condiciones para el desarrollo.

El siguiente factor que influyó de manera considerable, se presentó en las cuestiones de planeación del proyecto, el cual se relaciona directamente con la metodología de Proceso Unificado y MoProSoft. Se tuvo en cuenta la experiencia que se conocía a través de sitios especializados de Ingeniería de Software y de ingenieros con experiencia en esta rama; coinciden en que es difícil llevar una planeación o un Plan de Desarrollo del Proyecto que se realice al inicio de un proyecto y que el documento nunca cambie, máxime cuando se involucra tecnología nueva o vanguardista, es decir, el documento siempre estará en constante cambio. La experiencia que dejó la construcción del Sitio, fue el uso de Macromedia Flex, ya que fue implementada tecnología nueva la cual brindó la posibilidad de interactuar con ambiente J2EE. Por otra parte, se tuvo el problema de disponer de información limitada, además de que se invirtió una cantidad de tiempo en aprender y modificar la tecnología para implementar algunos de los Casos de Uso. Estos aspectos alteran los tiempos estimados y ocasionan constantes cambios en los tiempos planeados.

Otro de los factores que fueron saliendo a flote durante el desarrollo del sitio, fue el manejo de la tecnología flash. Un primer punto está relacionado con los diferentes navegadores Web, desde un principio se estableció que las mejores condiciones de navegación fueran para el navegador Internet Explorer en su version 6, por lo que se incluyó la característica de que el sitio fuera adaptable totalmente al tamaño de la ventana que el Usuario quisiera manejar. Esta característica permite visualizar el sitio en cualquier resolución del monitor, sin afectar la visibilidad del sitio. Sin embargo, manejar esta funcionalidad con otros navegadores, como Netscape o FireFox en todas sus versiones, genera conflictos, ya que el sitio se ve muy pequeño dentro del navegador. Por otro lado, con la instalación del Service Pack 2 de Windows XP, ocasiona un problema con los elementos flash, a menos que el usuario modifique el grado de protección de su sistema. Un punto importante fue que el manejo de flash impidió realizar las pruebas de carga y rendimiento con un software especializado, ya que ninguno de ellos soporta Flash; la utilización de este tipo de software hubiera garantizado la correcta funcionalidad del sitio.

RESULTADOS

Los factores descritos en los párrafos anteriores fueron los principales obstáculos para trabajar apegados al 100% a una metodología. En seguida, como parte de los resultados logrados, se maneja el punto correspondiente el Nivel de Capacidad obtenido.

Contemplando los resultados obtenidos, se considera que el proyecto alcanzó el propósito establecido mediante la ejecución de los procesos, desarrollarse dentro del Nivel 1 de Capacidad, el cual tiene por norma general: “realizar el proceso al 100%, sin importar como se haga, pero cumpliendo el propósito del proceso”.

En el Nivel 2 de capacidad fundamentalmente se pide que la administración de los procesos sea planeada, ajustada y supervisada, además que los productos sean controlados y mantenidos. Sin embargo en sólo algunos de los procesos para el desarrollo del sitio, se tienen las condiciones para alcanzar estos parámetros.

Relacionado con las cuestiones que se planteaban de la planeación, ésta resulta ser un requisito difícil de cumplir cuando se trabaja con tecnología nueva, por lo que el control y la planeación pueden establecerse teóricamente, pero en la práctica se deben realizar constantes adecuaciones. En la Figura 7.1 Nivel de Capacidad de los Procesos se representan los resultados obtenidos de la medición de los procesos.

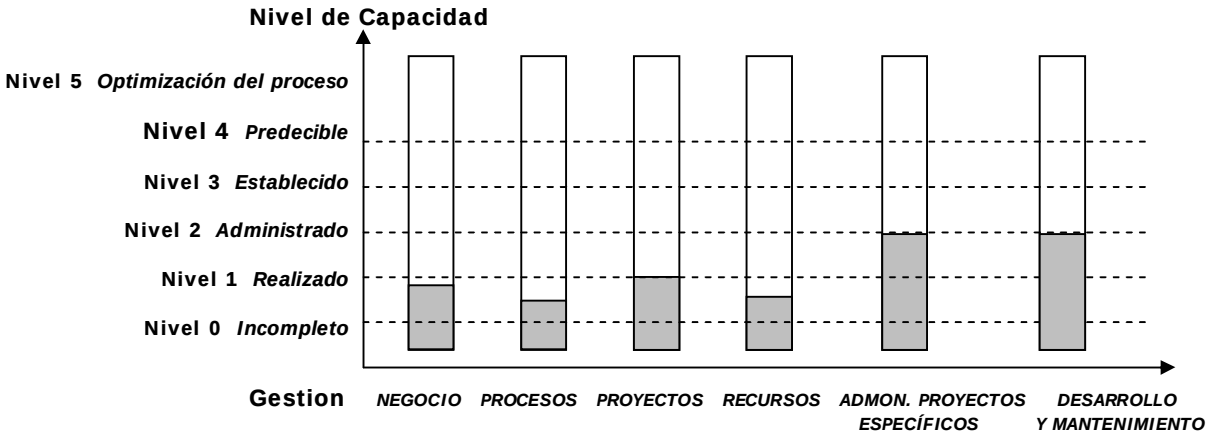


Figura 7.1 Nivel de Capacidad de los Procesos
Fuente: Elaboración propia.

Analizando los resultados, la gestión de Negocio está prácticamente en un Nivel 1; su objetivo principal es realizar los proyectos apegados a la filosofía que encierra el Plan Estratégico de la Organización. El proyecto del Sitio Web fue guiado por este documento, sin embargo para muchos de los integrantes del LTST no es conocido, por lo tanto el Nivel 1, en este proceso, no se logra al 100%.

La gestión de Procesos trató de seguir las bases que dicta MoProSoft, sin embargo para lograr el Nivel 1 se da por hecho que todos los procesos están definidos y establecidos, si es el caso, automáticamente están en un Nivel 3, sin embargo esto no se cumple para la organización.

En la gestión de Proyectos, se cumplieron las expectativas consistentes en cumplir con los objetivos trazados para la organización, por medio de proyectos internos. Por lo que se considera que se llegó al Nivel 1. Para el proceso de gestión de Recursos, se realizaron las tareas y productos que se podían generar, principalmente la Base del Conocimiento y formatos de solicitudes de recursos se considera que están a la mitad del Nivel. Los procesos de Administración de Proyectos y Desarrollo y Mantenimiento llegan a un Nivel 2, puesto que el proceso se llevó a cabo de manera planeada y supervisada, además de que sus productos de trabajo se consideran están mantenidos y controlados. De acuerdo a la Figura 7.1 podría obtenerse un promedio bajo la condición de que MoProSOft no establece un valor para los indicadores, por lo que existe cierta libertad de otorgar un puntaje:

- ***Gestión de Negocio: 0.8***
- ***Gestión de Procesos: 0.5***
- ***Gestión de Proyectos: 1***
- ***Gestión de Recursos: 0.6***
- ***Gestión de Administración de Proyectos: 2***
- ***Gestión de Desarrollo y Mantenimiento: 2***
- ***Promedio: 1.15***

Este promedio de 1.15 coincide con diferentes resultados realizados a empresas desarrolladoras de software, al comenzar a aplicar estos marcos de trabajo, por ejemplo en los resultados publicados en el documento ***"Estudio del nivel de madurez y capacidad de procesos de la industria de tecnologías de información en el área metropolitana de Monterrey, Nuevo León y el Distrito Federal y su área metropolitana", PROSOFT, 2004;*** y en una presentación de resultados sobre pruebas Controladas por parte de AMCIS el 31 de marzo del 2005.

ANEXO A

Este anexo muestra los documentos citados a lo largo del capítulo de Desarrollo del sitio Web. Cabe aclarar, que los documentos que a continuación se presentan, no corresponden al formato con el que se desarrolló este documento de tesis.

El inicio y término de un documento, se indica como se muestra a continuación:

Inicio de Documento

Contenido

Termino de Documento

Estándares

Los estándares definen los estilos a usar en cada uno de los elementos que pueden aparecer en los documentos del proyecto Sitio LTST. Primeramente se definió un documento que especifica cada uno de esos estilos.

Inicio de Documento

Proyecto WEB LTST
EstandarDocumentacion_v1.2.doc
Fecha de creación 20/feb/2005

Titulo ***APEEstandarDocumentacion***
Fecha (dd/mmm/aaaa) ***15/jul/2005*** Versión ***1.2***
Autor ***J. René Ramírez Hernández***
Descripción ***Aplicación de nuevo formato***

ESTÁNDAR DE DOCUMENTACIÓN

El documento provee de estándares para el manejo general acerca del formato que tendrán todos los artefactos creados para el proyecto.

Estructura de los documentos

Todos los documentos deberán tener índice en caso de que lo amerite. Inicialmente todos deberán tener un encabezado de página como se muestra:

WebLTST
Nombre_vx.x.doc
Fecha de creación

En donde "WebLTST" indica el título del proyecto que se está desarrollando. La línea inmediata, contendrá el nombre del documento, el cual contendrá la versión y la extensión, generalmente será .doc, la siguiente línea mostrará la fecha de creación del documento. Finalmente deberá llevar la imagen que hace referencia al proyecto.

Inicio del documento

Al inicio de cada documento deberá llevar una tabla como la que se muestra:

Titulo
Fecha (dd/mmm/aaaa) Versión
Autor
Descripción

Llenando en su espacio correspondiente información. El formato de fechas se estandarizó a 2 dígitos, las primeras 3 letras del nombre del mes con minúsculas y los cuatro dígitos del año en curso. Las versiones comienzan con la 1.0 y los pequeños cambios al documento ameritarán un incremento en la versión a 1.1, 1.2, 1.3,... así sucesivamente. Solamente los cambios radicales ameritarán un cambio a una versión 2.0.

Nombre del documento

El formato utilizado para nombrar los documentos generados en el proyecto WEB LTST es el siguiente:

ETIQUETA_vX.X.EXT	
ETIQUETA	Se utilizará las etiquetas dependiendo del nombre del documento de acuerdo a la tabla que se muestra a continuación.
_vX.X	Versión del documento.
EXT	Extensión de acuerdo al tipo de documento.

Títulos.

Irá solamente en la primera página. EL título es en Verdana de 11 puntos, negrita y centrado, además de ser todo con mayúsculas.

ESTÁNDAR DE DOCUMENTACIÓN

Pie de página.

El pié de página únicamente llevará el número correspondiente a la página del documento, con la salvedad que la última hoja del documento.

- 3 -

Subtítulos

Los subtítulos son en Verdana de 11 puntos, y alineado a la izquierda.

Subtítulo

Párrafos

Los párrafos serán en Verdana de 9 puntos, justificado, con interlineado sencillo.

Márgenes del documento

Superior:	2.5 cm	Izquierda:	2.5 cm
Inferior:	2.5 cm	Derecha:	2.5 cm
Encabezado:	1.25 cm	Desde el borde la página.	
Pie de página:	1.25 cm		

Tablas de contenido

Serán manejadas como tabla normal, centradas y ajustándose siempre al contenido de la tabla, sin embargo, las celdas que contengan o definan un nombre de renglón o columna tendrán un fondo gris (12.5%) y texto con tipo de letra Verdana a 9 puntos en negrita. El contenido de la tabla tendrá el formato señalado en el apartado de Párrafo.

Referencia de tablas y figuras.

Éste irá bajo el estilo “Cuadros, Tablas y Figuras”, el texto detallará de qué tipo se trate, si es esquema, un diagrama, una figura, etc., la cual deberá ir correctamente numerada y mediante un breve texto describirá su contenido.

Cuadros, tablas y figuras.

Viñetas.

Se hará uso de éstas, bajo el estilo Viñetas 1 como se muestra a continuación

- Uno
  - Contenido uno
- Dos
  - Contenido dos
- Tres
  - Contenido tres

Terminación del documento.

Cualquier documento tiene campos obligatorios que indican el “estado” en que se encuentra el documento; existen tres estados:

- En desarrollo
- Verificar y validar
- Listo

La función de cada uno de los estados es indicar en qué etapa se encuentra un documento, ***en desarrollo*** nos indica un documento que aún no se encuentra terminado su contenido, ***verificar y validar*** indica que el documento ha sido terminado, sin embargo debe ser revisado y autorizado por una persona capacitada para dar la aprobación de que el documento es correcto, por lo mismo, se deberá anexar un campo con el nombre de la persona que revisó el documento, cuando el documento este terminado y correctamente revisado se pasará al estado ***listo***.

Verificación y Validación

Estado

En desarrollo ☐
Verificar y validar ☐
Listo ☐

Validado por _____
Fecha xx/xxx/xxxx

Termino de documento

Minutas

Inicio de Documento

Titulo	
Fecha (dd/mmm/aaaa)	Versión
Autor	
Descripción	***Documento que detalla cada reporte de juntas, actividades y acuerdos tomados para el desarrollo del proyecto.***

MINUTA Nº.

Participantes:

Nombres

Faltantes:

Nombres

Actividades:

Verificación y Validación

Estado

En desarrollo ☐
Verificar y validar ☐
Listo ☐

Validado por _____
Fecha xx/xxx/xxxx

Termino de documento

Reportes de seguimiento

Inicio de Documento

Titulo ***Estándar Seguimiento Semanal de Actividades***
Fecha (dd/mmm/aaaa) | Versión |
Autor
Descripción ***Detalla el formato del reporte de las actividades realizadas durante la semana.***

SEGUIMIENTO SEMANAL DE ACTIVIDADES

Nombre:	Rol:	Flujo:
Fecha:	Período:	Semana:

Actividades

#	Actividad	Tiempo (horas)		Fecha final		Estado
		estimado	real	planeada	real	
1	Texto describiendo la actividad	XX :XX	XX :XX	dd/mm/aa	dd/mm/aa	texto
Totales:		XX :XX	XX :XX			

Productos

#	Nombre	Tamaño	Defectos		Estado
			encontrados	corregidos	
1		real / estimado	X	X	En corrección

Cambios

#	Documento	Descripción	Solicitante	Estado

Riesgos

#	Descripción	Plan de Contingencia	Estado

Resumen

Actividades
A tiempo:
Retrasadas:
Adelantadas:

Cambios
Solicitados:
Rechazados:
Realizados:
Regresados:

Riesgos
Encontrados:
Resueltos:
Postergados:

Verificación y Validación

Estado

En desarrollo ☐
Verificar y validar ☐
Listo ☐

Validado por _____
Fecha xx/xxx/xxxx

Termino de documento

Titulo ***Administración de la Base del Conocimiento***
Fecha (dd/mmm/aaaa) | Versión |
Autor
Descripción

ADMINISTRACIÓN DE LA BASE DEL CONOCIMIENTO WEB LTST

Introducción

El presente documento esta basado en el documento PlanAdministracionBaseConocimiento.doc. Se pretende que este Plan de Administración de la Base del Conocimiento sirva como un marco de referencia donde los integrantes del proyecto pueden consultar:

- Localización del repositorio del proyecto.
- Cuentas de acceso al repositorio del proyecto.
- Especificación de la estructura general del repositorio del proyecto.
- Identificación de los productos que deben ser almacenados en este depósito.
- Asignación de la nomenclatura a los productos.
- La ubicación de estos productos dentro del repositorio del proyecto.
- El procedimiento que se debe seguir en la organización para llevar a cabo cambios a los productos en la base del conocimiento.

Localización del repositorio del proyecto

El repositorio del proyecto se implemento en un servidor FTP cuya dirección IP es 132.248.231.197, utilizando un puerto no estándar, TCP 21. Esto permitirá un acceso sencillo, rápido y posible desde cualquier computadora en Internet, ventajas heredadas del propio protocolo de transmisión utilizado.

Se puede acceder al repositorio utilizando cualquier programa cliente de FTP o incluso navegadores WWW, debido a la facilidad de manejo de estos últimos, se recomiendan sobre los primeros. Si es el caso de acceder desde un navegador WWW, se puede utilizar el siguiente URL para ello: ftp://132.248.231.197: En el caso específico de Netscape Navigator se sugiere utilizar: ftp://username:password@132.248.231.197:21, donde username es la cuenta asignada a cada rol y password es la contraseña asociada a dicha cuenta; este último URL se debe utilizar con extrema precaución, ya que compromete la cuenta y su contraseña a solamente dar un vistazo en la pantalla o inspeccionar el caché de direcciones URL visitadas.

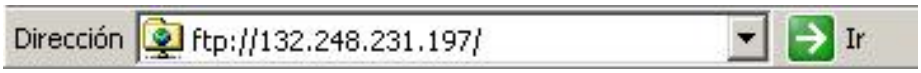


Figura 1: Ejemplo de acceso al URL del repositorio desde MS Internet Explorer y MS Windows Explorer

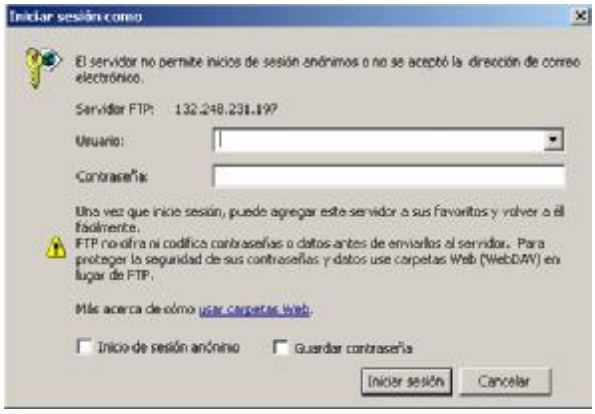


Figura 2: Ejemplo de ventana de validación para acceso al repositorio desde MS Internet Explorer y MS Windows Explorer

Cuentas de acceso al repositorio del proyecto

Para tener acceso al repositorio del proyecto se crearon cuentas de usuario con permisos de acceso de acuerdo al rol desempeñado, algunas de estas cuentas son individuales y otras genéricas. En la
Tabla 1 se listan las cuentas de acceso al repositorio y el recurso y rol asociado a cada una de ellas

Cuenta	Rol Asociado	Recurso
angelica	Gestión de negocios	Dra. Angélica del Rocío Lozano Cuevas
alex	Responsable de Gestión de Procesos / Responsable de P. E.	Luis Alejandro Guzmán C.
raul	Responsable Administración Mantenimiento de Software	Raúl Flores Morales
rene	Responsable Administración Base del Conocimiento	José René Ramírez Hernández

Tabla 1: Listado de cuentas de acceso al repositorio del proyecto

Creación de la Base de conocimiento de la organización

La estructura a primer nivel de detalle de la Base de conocimiento será la siguiente:

Figura 3: Estructura a primer nivel de detalle de la Base de Conocimiento

En el documento BaseDelConocimiento.doc, detalla en qué consisten cada uno de los repositorios.

Plan de configuración

Protocolo de resguardo

Se propone crear la estructura de la Base de Conocimiento en un servidor FTP accesible desde Internet, donde todos los miembros de la organización tendrán acceso a los documentos de acuerdo al rol(es) desempeñado. Actualmente se cuenta un servidor FTP en el siguiente URL ftp://132.248.231.197

La razón de usar un depósito en Internet es la de facilitar el acceso a la información de la organización a todos sus miembros.

Verificación y Validación

Estado

En desarrollo ☐
Verificar y validar ☐
Listo ☐

Validado por _____
Fecha xx/xxx/xxxx

Termino de documento

Referencias

1. Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Lenguaje Unificado de Modelado"; Adisson Wesley Iberoamericana, Madrid 1999

2. Rumbaugh, J.; Jacobson, I.; Booch, G.; "El Lenguaje Unificado de Modelado, Manual de Referencia"; Pearson Educacion S.A., Madrid 2000

3. Booch, G.; Rumbaugh, J.; Jacobson, I.; "El Proceso Unificado de Desarrollo de Software"; Adisson Wesley Iberoamericana, Madrid 1999.

4. Revista Software Guru, Año 1 No. 1 Enero-Febrero 2005. "Procesos de Software, Diversidad en Modelos".

5. Hanna Oktaba; "Introducción a estándares para la Ingeniería de Software y a SPICE. Estándares de la Ingeniería de Software". 10 de febrero 1999., Facultad de Ciencias, UNAM. ***URL: http://hp.fciencias.unam.mx/~ho/SPICE***

6. Hanna Oktaba; Presentación en Power Point "EvalProSoft y Pruebas Controladas", UNAM, AMCIS.
   URL: http://azul.iing.mxl.uabc.mx/eventos/moprosoft2005/

7. Hanna Oktaba (Director), Claudia Alquicira Esquivel, Angélica Su Ramos y otros; "Modelo de Procesos para la Industria de Software MoProSoft" Secretaría de Economía; Versión 1.1 Mayo 2003.

8. Macromedia. "Developing Rich Internet Applications with Macromedia MX 2004". August 2003.
   URL:http://www.macromedia.com/devnet/studio/whitepapers/rich_internet_apps.pdf

9. Macromedia® Flex™. "La solución de niveles de presentación para entregar aplicaciones dinámicas de Internet empresariales". Octubre 2004.
   URL:http://www.macromedia.com/es/software/flex/whitepapers/pdf/flex15_tech_wp.pdf

10. Ing. Luis A. Guzmán "Metodología Para Proyectos De Software Orientado A Objetos Basada En Cmm y El Proceso Unificado De Desarrollo De Software", Tesis Licenciatura, FI - UNAM, 2003.

11. LIBRO DE FLEX DE MACROMEDIA

12. Secretaría de Economía. "Programa para el Desarrollo de la Industria del Software (PROSOFT)".
    URL: http://www.economia.gob.mx/?P=1128

13. Laboratorio de Metodologías y Lenguajes. España.
    URL: http://ls.fi.upm.es/

14. Miguel Ángel Laguna. "2-Proceso", Departamento de Informática. Universidad de Valladolid. España,
    URL: http://www.infor.uva.es/~mlaguna/is2/2-Proceso.pdf

15. Secretaría de Economía, "Programa para el Desarrollo de la Industria del Software, PROSOFT", Junio 2003.
    URL: http://economia.gob.mx/pics/p/p1128/PPT.pdf

16. Ana Vázquez, "Estrategia propuesta para la emisión de la norma mexicana para la industria de software", AMCIS, Junio 2003.