UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

DESARROLLO DE UN SITIO WEB SIGUIENDO LA METODOLOGÍA TSPI

acompañante de un disco compacto

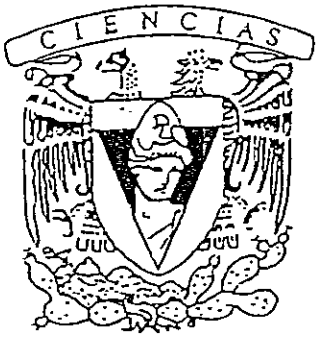
P R O Y E C T O

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

SERGIO RAMOS CHAVÉZ



DIRECTOR DEL PROYECTO:
DRA. HANNA OKTABA

JULIO 2002

M. EN C. ELENA DE OTEYZA DE OTEYZA
Jefa de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

"Desarrollo de un Sitio WEB Siguiendo la Metodología TSPi"

realizado por Ramos Chávez Sergio

con número de cuenta 09653528-2 , quién cubrió los créditos de la carrera de Ciencias de la Computación.
Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	Dra. Hanna Oktaba	
Propietario	M. en C. María Guadalupe Elena Ibargüengoitia González	
Propietario	Dr. Fernando Gamboa Rodríguez	
Suplente	M. en I. María de Luz Gasca Soto	
Suplente	M. en C. Paul Oswaldo Saldaña Nava	

Consejo Departamental de Matemáticas

Dra. Amparo López Gaona

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMATICAS

Agradecimientos

A Dios:
Por su ayuda incondicional.

A mis Padres:
Ma. de Lourdes Chávez Azuara y Sergio Ramos Pérez, por su soporte y aliento.

Al Director del Proyecto:
Dra. Hanna Oktaba por sus enseñanzas, paciencia, dedicación e impulso incansable para la terminación de éste trabajo.

A los Sinodales:
M. en C. María Guadalupe Elena Ibargüengoitia González, Dr. Fernando Gamboa Rodríguez, M. en I. María de Luz Gasca Soto, M. en C. Paul Oswaldo Saldaña Nava por su valioso tiempo y gran disposición.

A todos los que me ayudaron y alentaron a terminar el presente proyecto.

A todos, ¡Muchas Gracias!

Índice General

Índice de Figuras

Índice de Tablas

Introducción

El término Ingeniería de Software fue inicialmente difundido cuando se usó como nombre de un taller propuesto por la OTAN en 1968. Dicho taller fue usado para fijar la atención en los problemas de desarrollo de software. Desde entonces, la Ingeniería de Software se estableció como una área de la computación cuyos conocimientos se ocupan de los métodos usados en la elaboración del software.

La tecnología avanza a pasos agigantados, y la forma de pensar acerca de cómo desarrollar algún tipo de software ya no resulta tan sencillo, ya que los requerimientos esenciales que deben llevar los programas actuales son múltiples, para que puedan considerarse como aceptables y funcionales en todos los sentidos.

Los parámetros que nos puedan indicar en forma verídica el desarrollo y ejecución del software son de suma importancia, no sólo para la calidad del producto, sino también para la mejora de las prácticas en el desarrollo. Aunque el software nos de el resultado deseado, debemos indagar en el *-cómo-* y *-a costa de qué-* , esto con el propósito de:

- Evitar faltas en documentación que consecuentemente deterioren la calidad del producto
- Hacer reflexionar a los involucrados en el proyecto sobre sus prácticas de desarrollo.

Una de las propuestas más recientes de la Ingeniería de Software es Introduction to Team Software Process(TSPi)[1] , que no sólo abarca aspectos técnicos de desarrollo sino que incluye elementos de planeación, métricas y aseguramiento de calidad del software. El proceso de TSPi ha sido experimentado en cursos de la licenciatura en Ciencias de la Computación

[1] Watts S. Humphrey *Introduction to Team Software Process*^{SM} , Addison Wesley, 2000

(Facultad de Ciencias 2000-1 y 2000-2) y de la Maestría en Ciencia e Ingeniería de la Computación (2000-2) en la Universidad Nacional Autónoma de México por la Dra. Hanna Oktaba y la M. en C. Guadalupe Ibargüengoitia.

Por otra parte, el desarrollo de sitios web es una tendencia que actualmente está en aumento. Todo mundo quiere aprovechar la difusión que un sitio WEB puede lograr en Internet, éste como cualquier otro producto de software, requiere seguir las normas de una metodología que le permitan obtener un alto grado de calidad.

Es en este punto es donde el presente proyecto involucró el uso de TSPi como metodología de Ingeniería de Software para el desarrollo de un sitio WEB. A su vez, el sitio generado, actúa como una herramienta de consulta de la misma metodología de TSPi. El proyecto particular del sitio TSPi está enmarcado dentro de un proyecto general de creación de un sitio WEB con material de apoyo para la comprensión de procesos de la Ingeniería de Software. El objetivo del proyecto general es proporcionar material actualizado, enriquecido con ejemplos, y visualmente atractivo, que auxilie a los maestros y alumnos en los cursos relacionados con la Ingeniería de Software. También, el mismo material puede servir para la actualización de los profesionales del desarrollo de software.

Fijamos los objetivos del proyecto como siguen:

- Desarrollar un sitio WEB para la metodología TSPi. Su finalidad es la de auxiliar a maestros y alumnos en cursos de Ingeniería de Software, y en general a personas interesadas en la metodología.
- Seguir la metodología de TSPi para el desarrollo del sitio

Para la realización del proyecto del sitio TSPi, se contó con una serie de diagramas tipo UML que abstraen la metodología TSPi, con ejemplos de la metodología, y de un formato en español de los guiones de la misma. Los diagramas que abstraen la metodología son parte de una tesis de la maestría de Ciencia e Ingeniería de la Computación de la UNAM elaborada por el M. en C. Paul Saldaña. Los ejemplos son productos desarrollados por alumnos de la misma maestría durante el curso de Ingeniería de Software Orientada a Objetos. Los guiones son parte de una tesis de la misma maestría elaborada por la M. en C. Carmen Dolores Alvarez Sánchez.

Durante el desarrollo del proyecto se mantuvo una constante revisión por parte de la Dra Hanna Oktaba(asesor del proyecto) y del M en C. Paul

Saldaña(proveedor de información del sitio), quienes fumían como cliente, usuarios y en gran parte como supervisores de la calidad.

El presente trabajo está dividido en seis partes:

En el Capítulo 1 se presenta una visión general de los procesos de la Ingeniería de Software, así como de la noción de calidad en el software.

En el Capítulo 2 se hace referencia a la metodología de TSPi, explicando y mostrando su estructura general.

En el Capítulo 3 se describen las distintas tecnologías involucradas en la implementación del proyecto.

En el Capítulo 4 se efectúa una descripción general del desarrollo del sitio WEB TSPi con la metodología de TSPi y haciendo uso de la tecnología descrita en el Capítulo 3.

En el Capítulo 5 se enuncian las conclusiones obtenidas a partir del desarrollo del proyecto.

En el Anexo A se describe la presentación de la información generada para cada ciclo del proyecto, que se encuentra en el CD que acompaña al trabajo.

Capítulo 1

Ingeniería de Software

La Ingeniería de Software es una disciplina o área, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelve problemas de todo tipo. Hoy día es cada vez más frecuente la consideración de la Ingeniería de Software como una nueva área de la ingeniería, y el ingeniero de software comienza a ser una profesión implantada en el mundo laboral internacional, con derechos, deberes y responsabilidades que cumplir. De cualquier forma, un punto de vista que prevalece con respecto a la Ingeniería de Software, es que es un área en crecimiento y que aún no es totalmente madura[1].

Es necesario hacer notar que la Ingeniería de Software, está contenida en una categoría más amplia que toma en cuenta otras áreas como la Ingeniería de Hardware. Dicha categoría es conocida como Ingeniería de Sistemas de Cómputo, en la cuál, cada disciplina contenida procura un desarrollo ordenado de sistemas relacionados con el uso de las computadoras.

1.1 Definición

Existen varias definiciones propuestas para la Ingeniería de Software, entre las que podemos mencionar las siguientes:

> "Una aproximación sistemática, disciplinada y cuantificable aplicada al desarrollo, operación y mantenimiento del software; esto es, la aplicación de ingeniería de software".[1]
>
> "Promover el establecimiento de las bases y disciplinas para Software,

[1] del IEEE Std 610.12-1990,1991

similares a las encontradas en las ramas de ingeniería ya establecidas".[2]

"El uso de metodologías, herramientas y técnicas para resolver problemas prácticos que se presentan en la construcción, desarrollo, soporte y evolución del software".[3]

Para fines del presente trabajo se puede pensar que la Ingeniería de Software es la aplicación de métodos y conocimientos para crear soluciones de diseño, construcción, operación y mantenimiento al software y productos asociados.

1.2 Actividades del Ciclo de Vida del Software

El ciclo de vida del software es un concepto que abarca las etapas por las que un producto de software pasa desde que es concebido y hasta cuando ya no es usado [18]. De cada etapa en la evolución de un software determinado, resulta el desarrollo de o bien una parte del sistema, o algo asociado con el sistema, tal como un fragmento de especificación, un plan para probar el software, o un manual de uso.

Los sistemas de software vienen y van a través de una serie de episodios entre los que podemos considerar su concepción, desarrollo inicial, operación productiva, mantenimiento y retiro. Por más de una década, muchas descripciones del clásico ciclo de vida del software han aparecido [3] y usualmente incluyen algunas versiones de las actividades siguientes:

Iniciación / Adopción del Sistema

¿De dónde vienen los sistemas? En la mayoría de las situaciones, los nuevos sistemas reemplazan o complementan mecanismos de procesamiento existentes. Ya sea que éstos últimos hayan sido previamente automatizados, sean manuales o informales.

[2]del Comité de Ciencias de la OTAN ,Berlack H.R., Software Configuration Management, Wiley, 1992

[3]del Instituto para la Tecnología de la Información, NCR Canada, 1990.

Análisis y Especificación de Requerimientos

Identifica los problemas que un nuevo sistema de software se supone resolverá. Éstos requerimientos son desarrollados conjuntamente por los usuarios y desarrolladores del software. Hacemos notar que el éxito de un sistema es medido por que tan bien el software refleja los requerimientos enunciados, que tan bien los requerimientos reflejan las necesidades percibidas por el usuario, y que tan bien las necesidades percibidas por el usuario reflejan las necesidades reales[2].

Análisis y Especificación de Diseño

Identifica los componentes del software(funciones, flujos de datos, etcétera) y especifica la relación entre ellos, así como la estructura global del sistema. Mantiene un registro de las decisiones tomadas y genera una documentación base para la realización de la implementación(siguiente actividad). Dentro de ésta podemos encontrar las siguientes actividades:

- **Especificación Funcional:** Identifica y potencialmente formaliza los objetos de computación, sus atributos y relaciones, las operaciones que transforman estos objetos, las condiciones que restringen el comportamiento del sistema, y demás.
- **Partición y Selección (Construir vs. Reusar):**Dados los requerimientos y especificaciones funcionales, divide los sistemas en piezas manejables que denotan subsistemas lógicos, después determina si sistemas de software nuevos, existentes ó reusables corresponden a las piezas generadas.
- **Especificación de Diseño Detallado de los Componentes:** Se refiere a los aspectos sobre la construcción de algoritmos y de las estructuras de datos. Esto con la idea de generar a partir de datos proporcionados al sistema, las salidas previstas.

Implementación

Lleva las especificaciones del diseño a un código fuente operacional y valida su operación básica. Así como la especificación de los requerimientos es el *qué* para la etapa de diseño y el diseño es el *cómo*, la especificación de diseño es el *qué* para la actividad de implementación y la implementación es el *cómo*[2].

Integración de Software y Pruebas

Afirma y sostiene la integridad total del sistema de software. Esto lo logra verificando la consistencia y completud de los módulos implementados, verificando las interfaces e interconexiones contra sus especificaciones, y validando el desempeño del sistema y subsistemas contra los requerimientos.

Revisión de Documentación y Liberación del Sistema

Empaca y racionaliza una descripción obtenida a lo largo del desarrollo del sistema en documentos sistemáticos y guías de usuario. Esto con la finalidad de diseminar y dar soporte al sistema.

Mantenimiento del Software

Ve por la correcta operación de un sistema en su ambiente destino. Esto lo logra efectuado anexos funcionales, reparaciones, mejoras en el desempeño y conversiones. Es importante remarcar que a veces el costo del mantenimiento de un sistema, alcanza el mismo valor del costo del desarrollo del sistema e incluso lo puede llegar a rebasar.

1.3 Modelos de Ciclo de Vida del Software

La evolución de los modelos de software se remontan a los primeros proyectos de desarrollo de grandes sistemas. El propósito aparente de éstos modelos de ciclo de vida del software, fué el de proveer un esquema abstracto descriptivo del desarrollo de los sistemas. Tal esquema podría entonces servir como la base de la planeación, organización, coordinación, documentación y dirección de las actividades que desarrollan el producto[4].

Un modelo de ciclo de vida de software puede pensarse como una caracterización descriptiva o prescriptiva de la evolución del software. Es más fácil articular un modelo de ciclo de vida prescriptivo para indicar el cómo un sistema de software debe ser desarrollado. Esto es posible debido a que la mayoría de tales modelos son intuitivos. Lo anterior significa que muchos detalles del desarrollo de software pueden ser ignorados, vagamente contemplados, o generalizados. Desde luego que con éstos, se genera cierto interés por su robustez cuando se desarrollan distintos tipos de sistemas bajo distintas circunstancias. Los modelos de ciclo de vida descriptivos, por otro lado,

caracterizan cómo los sistemas de software son actualmente desarrollados. Estos son menos comunes y más difíciles de articular por una razón: se debe observar y recolectar información a través del desarrollo del software durante un tiempo que usualmente es medido en años.

1.3.1 ¿Cómo pueden ser usados los modelos de Ciclo de Vida del Software?

Algunas formas de usar los modelos de ciclo de vida son:

- para organizar, planear, dirigir, calendarizar, y controlar el trabajo de un proyecto de software basándonos en el tiempo disponible, espacio de trabajo, y medios computacionales;
- como lineamientos prescriptivos sobre la documentación que hay que producir para la liberación y entrega del software;
- como marcos de trabajo para analizar o estimar patrones de distribución y consumo de recursos durante el ciclo de vida del software;
- como base para determinar que herramientas y metodologías de Ingeniería de Software serán las más apropiadas para soportar distintas actividades del ciclo de vida;
- como una base para conducir estudios empíricos para determinar que afecta la productividad, costo, y calidad del software;

1.3.2 Algunos Modelos de Ciclo de Vida del Software

A continuación se describen algunos modelos:

Modelo Cascada

Éste es el modelo clásico de ciclo de vida del software, fue originalmente documentado por Royce en el año 1970. El modelo de cascada ve el desarrollo como una secuencia simple de fases en donde cada fase tiene un conjunto de metas bien definidas y las actividades entre cualquier fase contribuye a satisfacer las metas de las fases subsiguientes. El orden lineal, ayuda a discernir con facilidad el principio y fin de cada fase. Una presentación gráfica del modelo se puede observar en la Figura 1.1

Principios básicos del Modelo de Cascada:

- Definición del Plan del Proyecto antes de comenzar con éste.
- Definición del comportamiento externo del sistema deseado.
- Documentación de los resultados de cada actividad.
- Diseño del sistema previo a la codificación.

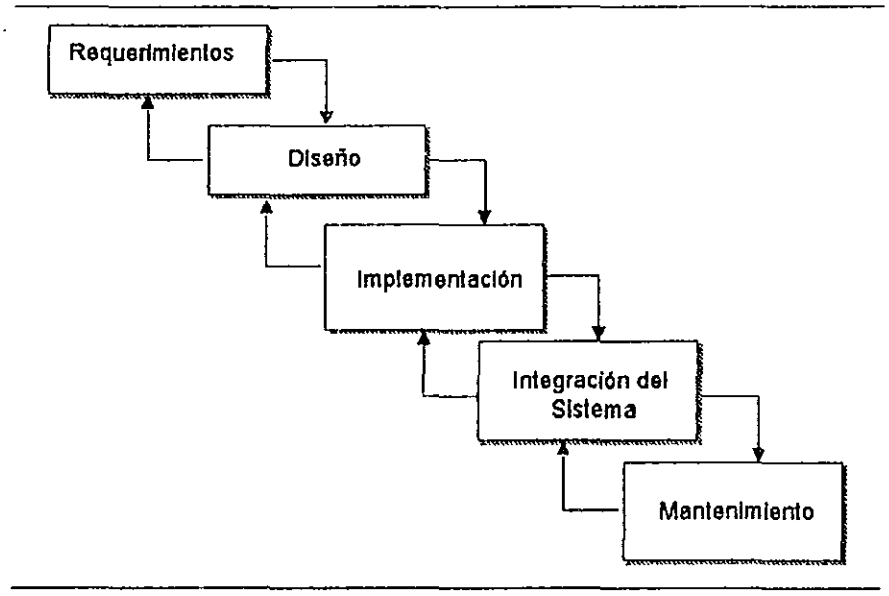


Figura. 1.1: Modelo de Ciclo de Vida Cascada.

Modelo Incremental

Los riesgos asociados con el desarrollo de sistemas largos y complejos son enormes. Una forma de reducir los riesgos es construir sólo una parte del sistema, reservando otros aspectos para niveles posteriores. El desarrollo incremental es un proceso de construcción en el que se van incrementando subconjuntos de requerimientos del sistema.Se puede observar una ilustración de lo anterior en la Figura 1.2

Notamos que el desarrollo incremental es totalmente compatible con el modelo cascada. El desarrollo incremental no demanda una forma específica de observar el desarrollo de algún incremento. Así, el modelo cascada puede ser usado para administrar cada esfuerzo de desarrollo. La Figura 1.3 nos muestra la unión del Modelo Inremental con el Modelo en Cascada.

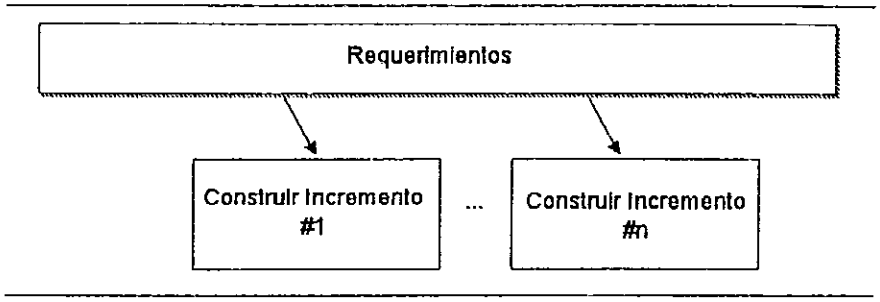


Figura. 1.2: Modelo de Ciclo de Vida Incremental.

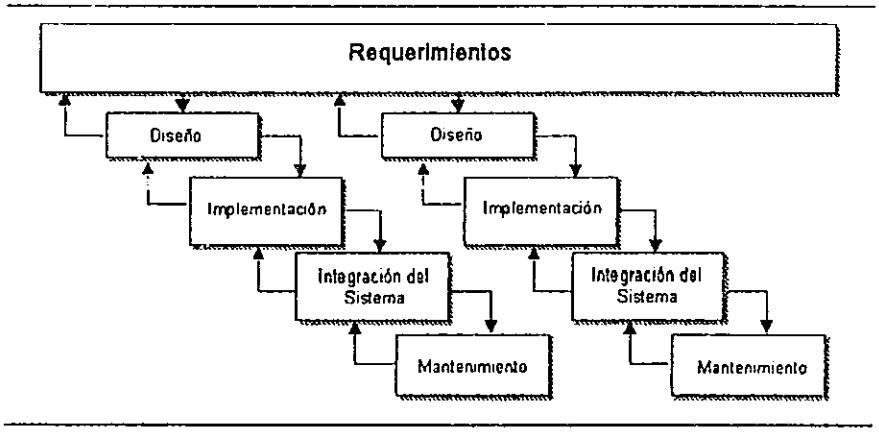


Figura. 1.3: Modelo de Ciclo de Vida Incremental con desarrollo en cascada de los incrementos.

El modelo de desarrollo incremental provee algunos beneficios significativos para los proyectos:

- Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande
- Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos
- Si un error importante es encontrado, sólo la última iteración necesita ser descartada
- Reduciendo el tiempo de desarrollo de un sistema (en este caso en incremento del sistema) decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo

- Los errores de desarrollo realizados en un incremento, pueden ser arreglados antes del comienzo del próximo incremento

Modelo Evolutivo

Como el modelo incremental, el modelo evolutivo (algunas veces denominado como prototipado evolutivo) construye una serie de grandes versiones sucesivas de un producto. Sin embargo, mientras que la aproximación incremental presupone que el conjunto completo de requerimientos es conocido al comenzar, el modelo evolutivo asume que los requerimientos no son completamente conocidos al inicio del proyecto. Ver Figura 1.4.

En el modelo evolutivo, los requerimientos son cuidadosamente examinados y sólo esos que son bien comprendidos son seleccionados para el primer incremento. Los desarrolladores construyen una implementación parcial del sistema que recibe sólo estos requerimientos.

El sistema es entonces generado, los usuarios lo usan y proveen retroalimentación a los desarrolladores. Basada en esta retroalimentación, la especificación de requerimientos es actualizada, entonces una segunda versión del producto es generada y desplegada. El proceso se repite hasta alcanzar un producto que satisfaga ciertas metas.

Notamos que el modelo evolutivo es completamente compatible con el modelo cascada. El modelo evolutivo no demanda una forma específica de observar el desarrollo de algún incremento. Así, el modelo cascada puede ser usado para administrar cada esfuerzo de desarrollo. Obviamente, el modelo incremental y evolutivo pueden ser combinados también.

Todo lo que uno tiene que hacer es construir un subconjunto de requerimientos conocidos (incremental), y comprender que muchos nuevos requerimientos es probable que aparezcan.Ésto durante la generación del sistema.

El desarrollo de software en forma evolutiva requiere un especial cuidado en la manipulación de documentos, programas, datos de prueba, etcétera, creados para distintas versiones del software. Cada paso debe ser registrado, la documentación debe ser recuperada con facilidad, los cambios deben ser efectuados de una manera controlada.

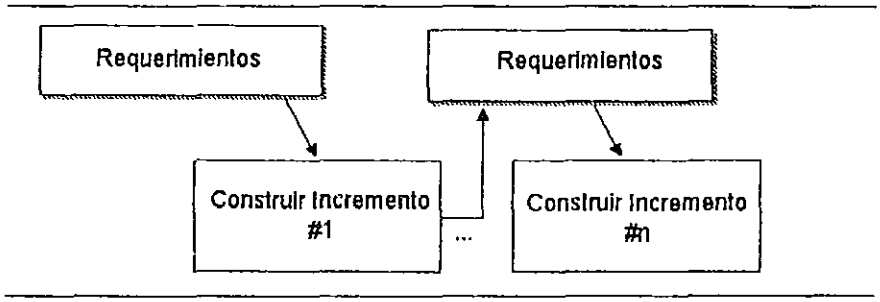


Figura. 1.4: Modelo de Ciclo de Vida Evolutivo.

Modelo de Espiral

Éste modelo fue desarrollado por Boehm[4] con el objetivo de combinar las ventajas de los modelos en cascada y evolutivo. En éste, se hace un tratamiento explícito del riesgo del proceso. Aquí se trata de desarrollar incrementalmente el proyecto, dividiéndolo en varios subproyectos. Uno de los puntos más importantes del proceso es concentrarse primero en los aspectos más críticos del proyecto. La idea es primero definir e implementar las características más importantes, y con el conocimiento adquirido para hacerlo, volver hacia atrás y reimplementar las características siguientes en pequeños subproyectos.

El modelo tiene la forma de una espiral, pudiéndose establecer una relación entre cada giro y una fase de desarrollo(ver Figura 1.5). Cada vuelta de la espiral se divide en cuatro sectores:

- **Definición de Objetivos:** Identificación de los objetivos específicos de cada fase
- **Evaluación y Reducción de Riesgos:** Identificación de los riesgos y solucion de éstos en cada fase.
- **Desarrollo y Validación:** Elección de un modelo adecuado para la siguiente fase del desarrollo.
- **Planificación:** Se revisa el estado del proyecto y se planifica el siguiente giro.

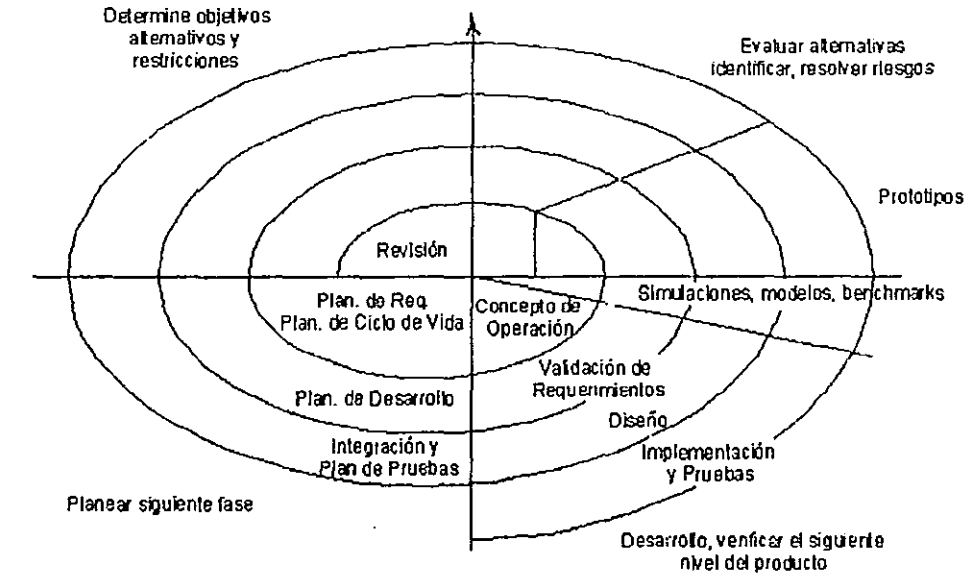


Figura. 1.5: Modelo de Ciclo de Vida Espiral.

1.4 Calidad del Software

La idea central que persiguen los métodos descritos en el presente trabajo, van orientados a una meta: producir software de calidad. Pero, ¿qué es software de calidad?

La calidad de un producto dado es algunas veces definida como

> "la totalidad de características [del producto o servicios] que se sustentan en su habilidad de satisfacer las necesidades enunciadas o implicadas".[4]

Pensando en software, un software de calidad podríamos pensarlo como

> "la concordancia de las necesidades funcionales y de rendimiento establecidos con el cliente, con los estándares de desarrollo documentados y con las características implícitas que se espera de todo software desarrollado".

[4]De Quality-Vocabulary, (ISO 8402:1986, note 1)

No hay duda de que la definición anterior de software de calidad puede ser ampliada o modificada. Incluso no tendría fin una discusión sobre una definición final. Para los propósitos de éste trabajo la anterior definición alude a cuatro puntos importantes:

1. El software debe satisfacer al cliente en todos sus aspectos
2. Los requisitos del software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad
3. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería del software. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
4. Existe un conjunto de requisitos implícitos que a menudo no se mencionan(como ejemplo, está el deseo de un buen mantenimiento). Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software queda en entredicho.

La noción de calidad de software no es tan simple como pareciera. Para cualquier producto de software, existen varias cualidades relevantes a ser discutidas y determinadas antes de empezar a generarse. Los atributos de calidad pueden estar presentes o ausentes, o bien pueden estar presentes en cierto grado. La ingeniería de software necesita, antes que nada, determinar el propósito real del software, que es el punto base a tener en mente: Las necesidades del cliente vienen primero, y éstas incluyen ciertos niveles de calidad, no sólo de funcionalidad. Entonces el ingeniero de software tiene la responsabilidad de esclarecer las necesidades de calidad que podrían a simple vista no estar contempladas y discutir la importancia y la dificultad de lograrlas.

1.4.1 Procesos de Soporte para la Calidad de Software

La Ingeniería de Software emplea múltiples procesos de soporte para examinar y garantizar la calidad en los productos. Estos procesos se aseguran de que las actividades de la Ingeniería de Software requeridas por el proyecto, sean alcanzadas. Dos de los procesos de soporte que están muy íntimamente relacionados con la calidad del producto, son la Garantía de Calidad de Software(SQA) y la Validación y Verificación($V\&V$)[6]. Ambos estimulan la calidad encontrando posibles problemas. Pero difieren un tanto en su énfasis.

Los procesos SQA y V&V proveen un manejo sobre la calidad de los productos en cada fase de su desarrollo o mantenimiento. El manejo mencionado viene de la visión de datos y medidas producidos de la aplicación de evaluaciones sobre los productos generados y en desarrollo, dentro de cualquier etapa del ciclo de vida del software. Incluso en algunos casos, las tareas asignadas para la garantía de calidad, captura de datos y medidas son realizadas por una organización independiente de la organización que tiene a cargo el proyecto de software. Esto es con el objetivo de proveer un alto grado de objetividad en términos de calidad.

Proceso de Garantía de Calidad del Software(SQA)

El proceso de SQA es un *diseño de acciones planificado y sistemático* [5], que proveen la certeza de que los productos de software y los procesos en el ciclo de vida del proyecto satisfagan sus requerimientos. La idea es asegurar que el problema sea clara y adecuadamente enunciado, además que los requisitos de la solución sean definidos y expresados de igual manera. El proceso SQA busca mantener la calidad a través del desarrollo y mantenimiento del producto mediante la realización de una variedad de actividades en cada etapa. De esto resulta una identificación temprana de problemas, que son prácticamente inevitables en cualquier desarrollo complejo.

El grupo que conforma el proceso de SQA, sirve como representación del cliente dentro de la organización que desarrolla el proyecto. Esto significa que los encargados del SQA, deben ver al software desde el punto de vista del cliente. ¿Se ha desarrollado el proyecto de acuerdo con los estándares preestablecidos? ¿Se satisfacen los aspectos de calidad acordados? El grupo a cargo del proceso de SQA intenta responder a éstas y otras cuestiones para asegurar que se mantiene la calidad en el proyecto.

Existen siete actividades principales dentro del proceso SQA, las cuales son:

1. Aplicación de métodos técnicos
2. Revisiones de técnicas formales
3. Prueba del proyecto
4. Ajuste a los estándares

5. Control sobre los cambios

6. Mediciones

7. Registro y generación de informes

La calidad del software debe estar diseñada para el producto, no es algo impuesto posteriormente. Por tal motivo, el proceso SQA inicia realmente con un conjunto de herramientas y métodos técnicos que ayudan a los analistas a conseguir una especificación y diseño de alta calidad.

Proceso de Verificación y Validación(V&V)

El proceso V&V determina si los productos de un desarrollo o mantenimiento dado, cumplen con las necesidades de dicha actividad, y con las impuestas por actividades previas. Buscando que el producto de software final cumpla con su uso especificado y con las necesidades establecidas de los usuarios.

La Verificación intenta asegurar que el producto está construido correctamente, en el sentido de que, el producto generado en una etapa, cumpla los requisitos impuestos sobre él en etapas previas. Las actividades de Verificación van a examinar un producto específico, es decir, el resultado de un proceso, y proveen evidencia objetiva de que los requisitos especificados han sido cumplidos. Los "requisitos especificados" se refieren a los requisitos del producto examinado, relativos ó relacionados con producto del cuál fue derivado. Por ejemplo, el código es examinado relativo a los requisitos de un diseño, o los requerimientos de software son examinados relativos a los requerimientos del sistema.

La Validación intenta asegurar que el producto correcto está construido, esto es, que el producto generado en una etapa satisface el uso para el cual fue destinado. Las actividades de la Validación examinan un producto para proveer evidencia objetiva de que cumple un uso específico deseado. La Validación confirma que el producto puede llevarse de regreso a los requerimientos del sistema de software y satisfacerlos. La Validación incluye una planeación para pruebas de sistema más o menos en paralelo con el proceso de requerimientos del software. Éste aspecto de la Validación muchas veces sirve como parte de una actividad de Verificación de los requerimientos. Mientras que algunas comunidades separan completamente la Verificación de la Validación, las actividades de cada una pueden servir a

la otra.

Ambos procesos de validación y verificación comienzan tempranamente en el desarrollo y en el proceso de mantenimiento. El proceso V&V proveen una examinación de todo producto obtenido, relativa a su predecesor inmediato y a los requisitos del sistema que debe cumplir.

Capítulo 2

TSPi como Metodología de Desarrollo de Software

Para que la Ingeniería de Software pueda evolucionar en una disciplina más madura, es necesario que los procesos de desarrollo de software cumplan con ciertas metas. Podemos mencionar por ejemplo:

- volverse mejor estructurados
- hacer mejores predicciones
- basarse más en estándares y prácticas establecidas
- recolectar datos para analizar y evaluar el proceso

Debido al creciente interés en las técnicas de desarrollos de los procesos del software. Se han generado diversas metodologías para trabajar sobre diversos medios. Algunos ejemplos de los procesos son:

- procesos para las organizaciones, como CMM, SPICE, ISO9001;
- procesos para equipos ó pequeñas organizaciones, como TSP;
- procesos individuales, como PSP.

En éste punto podemos decir que TSPi(Introduction to Team Software Process) es una propuesta para desarrollo en equipos de proyectos de software relativamente pequeños. Su intención es primordialmente didáctica. Ésta metodología está influenciada por el modelo en espiral. TSPi ofrece guiones y formas para guiar al equipo de trabajo en la aplicación de conocidos procesos del desarrollo en equipo exitosos. Éste capítulo revisa la metodología TSPi basándose en gran parte, en el libro *Introduction to Team Software Process*^{SM} de Watts S. Humphrey[11].

2.1 La Relación entre PSP, TSP y TSPi

Desde comienzos de 1989, Watts S. Humphrey trabajó en un intento por usar procesos para grandes proyectos en pequeños proyectos. Durante casi cuatro años de trabajo, recolectó datos en tamaño, tiempo y defectos, y desarrolló una metodología personal de desarrollo de software[7], conocida como Personal Software Process(PSP). Posteriormente dicha metodología fue empleada para entrenar a otras personas en el desarrollo de software, los resultados de éstos estudios pueden ser encontrados en The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers [8].

El diseño de PSP está basado en los siguientes principios de planeación y calidad[10]:

- Todo ingeniero de software es diferente; para ser más efectivo, los ingenieros deben planear su trabajo y deben basar sus planes en sus propios datos personales.
- Para mejorar consistentemente sus desempeños, los ingenieros deben personalmente usar procesos medibles y bien definidos.
- Para producir productos de calidad, los ingenieros deben sentirse personalmente responsables por la calidad de sus productos. Los productos superiores no son producidos por error; los ingenieros deben esforzarse por hacer un trabajo de calidad.
- Es menos costoso encontrar y reparar los defectos tempranamente en un proceso, que tardíamente.
- Es más eficiente prevenir defectos que encontrarlos y repararlos.
- La manera correcta de hacer un trabajo, es siempre la más rápida y menos costosa.

Según PSP, para llevar a cabo un trabajo de ingeniería de software de la manera adecuada, los ingenieros deben planear antes de comenzar y deben usar procesos definidos para planear el trabajo. Para entender su desempeño personal, deben medir el tiempo en cada paso del trabajo, los defectos que inyectan y remueven, y los tamaños de los productos que generan. Para que se puedan producir productos de calidad consistentemente, los ingenieros deben planear, medir y llevar un registro de la calidad del producto. Además deben centrarse en la calidad desde el principio de un trabajo. Finalmente,

ellos deben analizar los resultados de cada trabajo y usar los resultados para mejorar sus procesos personales.

Humphrey reconoció el éxito que se había logrado con PSP en los salones de clase, y era claro que los desarrolladores podían usar el PSP para tener un mayor control de sus procesos personales sin afectar su productividad[8].

De cualquier forma, no era muy claro el uso de PSP dentro de una pequeña empresa. Al parecer, si algún ingeniero era entrenado con PSP, y no tenía el medio adecuado para usar la metodología, tarde o temprano la abandonaba[9]. Consecuentemente, Humphrey desarrollo el Team Software Process(TSP), como una respuesta operacional al problema de implantación de PSP a pequeñas empresas. El TSP fue introducido desde entonces en los medios laborales así como en los académicos.

Es en éste punto donde TSPi se propuso como un esquema introductorio y didáctico, para el aprendizaje de TSP. Los cursos con TSPi han sido enseñados en algunas universidades(incluyendo entre éstas a la UNAM, siendo sus principales promotores la Dra. Hanna Oktaba y M. en C. Guadalupe Ibargüengoitia), y las experiencias tempranas indican que los estudiantes y profesores la encuentran útil. De cualquier manera, aún no hay estudios formales al respecto.

2.2 Objetivos de TSPi

El TSPi está basado en el PSP y fue diseñado para ayudar a los equipos de software o pequeñas organizaciones en la aplicación principios de ingeniería y de administración de desarrollo de software.

El TSPi es un subconjunto de TSP, y sus objetivos son:

- proveer guía y soporte para el desarrollo exitoso de un proyecto modesto a través de varios ciclos de desarrollo incremental;
- proveer un balance en el énfasis sobre proceso, producto, y equipo de trabajo;
- sugerir una guía sobre las características y problemas del trabajo en equipo;

- procurar un marco de trabajo medible y definido para el desarrollo de un producto de software.

2.3 Principios de TSPi

TSPi está basado en los siguientes cuatro principios básicos:

1. Aprender es más efectivo cuando se sigue un proceso definido y se obtiene una rápida retroalimentación. Los guiones y formas de TSPi, proveen un marco de trabajo definido, medido, y repetible para equipos de Ingeniería de Software. El TSPi provee una rápida retroalimentación del desempeño, porque el equipo produce el producto en varios ciclos cortos de desarrollo y evalúa los resultados después de cada ciclo.

2. Los grupos de trabajo productivos requieren una combinación de metas específicas, soporte en el medio de trabajo, y un entrenamiento y liderazgo capaces. La meta del proyecto es construir un producto funcional. TSPi provee el ambiente de soporte, un líder de equipo que es un miembro del mismo y el instructor provee el entrenamiento.

3. Cuando alguien se enfrenta con problemas del proyecto en desarrollo y ha sido guiado hacia soluciones efectivas, aprecia los beneficios de las prácticas de un desarrollo sólido. Sin la guía precisa de TSPi, se podría invertir un tiempo considerable en definir las propias prácticas, métodos y roles.

4. La instrucción es más efectiva cuando se construye sobre conocimiento previamente establecido. Se ha tenido un gran cúmulo de experiencia con equipos de software y cursos con equipos de software. TSPi se basa en éstos fundamentos.

Además, el diseño de la metodología de TSPi está basado en las siguientes siete decisiones:

1. Proveer un marco de trabajo simple que se base en el PSP.

2. Desarrollar productos en varios ciclos.

3. Establecer métricas y estándares para calidad y desempeño.

4. Proveer métricas precisas para equipos y estudiantes.

5. Usar la evaluaciones de rol y de equipo.

6. Requerir disciplina en el proceso.

7. Proveer una guía en los problemas del trabajo en equipo.

2.4 Estructura de Desarrollo de TSPi

La estructura de desarrollo de TSPi se basa en la repetición de ciclos de desarrollo, hasta alcanzar un producto final. Durante un curso, generalmente no se emplean más de tres ciclos, aunque se pueden efectuar un número mayor dependiendo de las necesidades del proyecto. El equipo sigue al TSPi a través de siete pasos de proceso en cada ciclo: lanzamiento, estrategia, planeación, requerimientos, diseño, implementación, pruebas y postmortem. En la Figura 2.1 se puede observar un esquema de la estructura de desarrollo de TSPi.

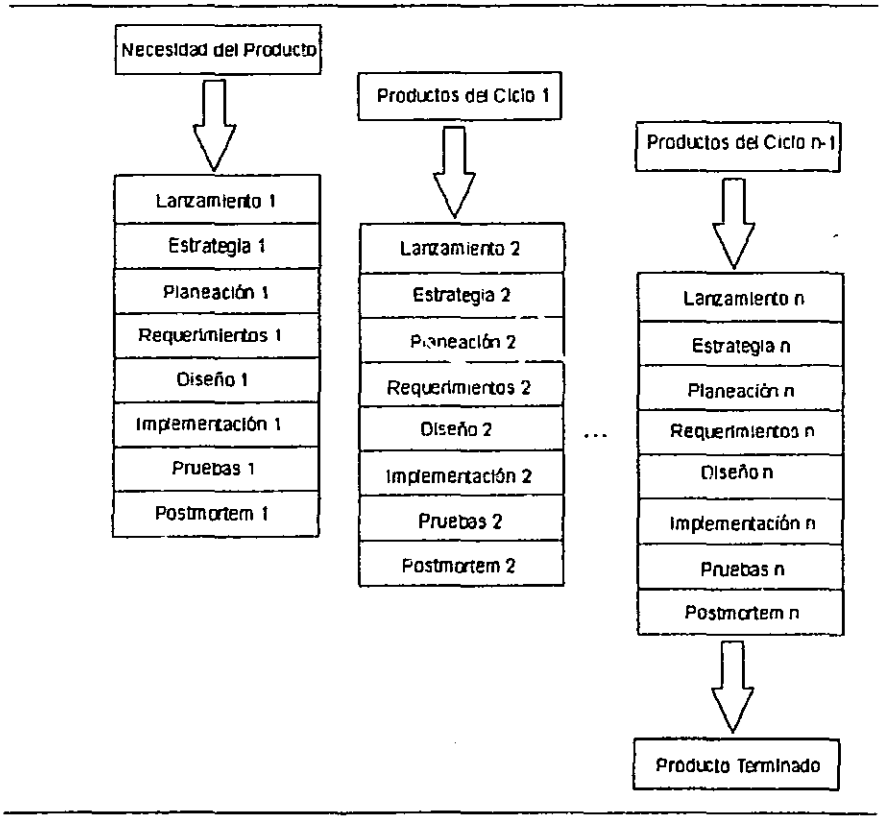


Figura. 2.1: Estructura de Desarrollo TSPi.

2.5 Descripción de los Roles de TSPi

A continuación se presentan los roles involucrados en la metodología de TSPi, y una breve descripción de su participación en la metodología.

Líder del Equipo: Debe mantener un equipo efectivo. Motiva a los integrantes del equipo a cumplir los compromisos adquiridos. Es el encargado de dirigir las reuniones de equipo, y se reúne semanalmente con el instructor para darle un informe de trabajo semanal.

Administrador de Desarrollo: Guía al equipo hacia el desarrollo de un producto superior. Prácticamente él dirige los desarrollos técnicos del equipo.

Administrador de Planeación: Su meta principal es la de producir una completa, precisa y puntual planeación del proyecto. Éste está involucrado en los procesos de planeación que sugiere la metodología.

Administrador de Calidad y Proceso: Encargado de guiar al equipo en su búsqueda por la calidad en los procesos y en el producto. Involucrado en el desarrollo del plan de calidad, y moderador de toda inspección realizada.

Administrador de Soporte: Asegura el abastecimiento del equipo con herramientas y métodos necesarios para el desarrollo adecuado del proyecto. Éste se encarga del esquema de control de cambios del producto.

2.6 Descripción de Fases en los Ciclos de TSPi

A continuación se presenta una breve descripción de las pasos o fases de cada ciclo perteneciente a TSPi.

Lanzamiento

En ésta fase, se define que miembros del equipo asumirán los distintos roles propuestos por TSPi. Por ser ésta la etapa inicial, se fijan las metas que el equipo piensa seguir durante el ciclo en curso. Las metas del equipo establecen el marco de trabajo para la estrategia y la planeación. Éstas,

en turno, proveen las bases de todo lo que el equipo hará en el ciclo. Básicamente hay 3 tipos de metas que TSPi sugiere definir, estas son el producir un producto de calidad, desarrollar un proyecto productivo y bien administrado, y por último terminar a tiempo.

Estrategia

Ésta fase contempla para el primer ciclo de cada proyecto, la definición global de un tentativo esquema de desarrollo de todo el producto. Dicho esquema incluye un estimado de la cantidad de ciclos y metas a realizar en cada uno de ellos. En los ciclos de desarrollo siguientes, se regulariza el esquema propuesto. Como resultado de ésta fase, se obtiene una estrategia de desarrollo del producto, una evaluación de riesgos, y un mecanismo de manejo y control de cambios del proyecto.

Planeación

La planeación en una actividad de suma importancia en TSPi, pues se trabaja de una manera más efectiva. Después de que se ha pensado en el trabajo a realizar, se sabe que es lo que hay que hacer y cuando hacerlo. Para la etapa de Planeación, se hace un estimado del tamaño y tiempo requerido de todos los productos del ciclo correspondiente a esa planeación. También se genera un plan de calidad que determina las metas a ser alcanzada por los productos.

Requerimientos

En ésta fase se realiza una especificación de la funcionalidad del producto. Dicha especificación se valida junto con el cliente, pues ésta debe reflejar lo que el cliente espera del producto. Además, la especificación genera una concepción común en el equipo de que el lo que se va a construir. También se genera un plan de evaluación del producto para probarlo una vez realizado, y asegurar su correcto desempeño.

Diseño

Aquí se decide cómo se va a construir el software del ciclo en curso. Se genera un diseño global pero completo de los componentes a desarrollar y de su integración, éste diseño es conocido en TSPi como Diseño de Alto Nivel. Entre las meta de éste diseño esta la de dar a los ingenieros una idea general de las partes del sistema y su interacción. Otro aspecto abordado en el diseño es la definición de estándares sobre el producto. Esto con la intención, entre otras cosas, de hacer organizado y entendible para los miembros del equipo, el trabajo generado. Por último, se genera un plan de integración que revisa las conexiones entre los componentes a desarrollar.

Implementación

La implementación intenta mapear las ideas del diseño en un código funcional. Dentro de la fase de implementación, TSPi maneja inicialmente un diseño detallado, en el que se hace una descripción detallada de los módulos propuestos en el diseño de alto nivel. Posterior a éste, se genera el código de cada módulo en el lenguaje de programación elegido. Una vez terminada la codificación, se efectúan pruebas sobre los módulos para dejar un escenario listo para la siguiente fase.

Integración y Plan de Pruebas

Durante ésta fase, se efectúa la integración de los módulos del sistema. Siguiendo las indicaciones generadas en el plan de integración generado en el diseño. También en ésta fase se conduce un sistema de pruebas del producto. A pesar de que se van a corregir los errores detectados durante éstas pruebas, la mayoría de los defectos se tuvieron que haber encontrado y corregido en etapas anteriores. El tiempo invertido en éstas pruebas es un indicativo de que tanta calidad fue obtenida durante el proceso. Finalmente, en ésta fase se genera la documentación del usuario. TSPi señala que la documentación es una parte clave del producto de software, y que ésta no debe estar centrada en el diseño del producto sino en las necesidades del cliente.

Postmortem

En ésta última fase, TSPi propone el desarrollo de un reporte de los datos obtenidos del ciclo contra los datos planeados del mismo. TSPi también ofrece mecanismos para hacer reflexionar al equipo y a cada miembro

del mismo sobre su desempeño. La meta de las propuestas anteriores es la de detectar que fue lo que estuvo bien y que fue lo que estuvo mal, para entenderlo e intentar mejorar el trabajo la siguiente vez.

Capítulo 3

Tecnología Empleada en el Desarrollo del Sitio WEB TSPi

En este apartado nos referimos a la tecnología usada en del desarrollo del presente proyecto. Por tecnología se entienden los recursos y herramientas particulares empleados durante la creación del software.

Entre las tecnologías mencionaremos una descripción de Internet, que fué el medio en el cuál el proyecto fué desarrollado y es actualmente consultado. También se hará mención de la Programación Orientada a Objetos, que fué el paradigma de programación empleado en el diseño e implementación. Se incluye una descripción de Java, como lenguaje de programación con el paradigma antes mencionado, y que fué empleado en la implementación. Por último, se hace referencia a las tecnologías empleadas para el acceso de la información persistente, inicialmente el proyecto empleo una Base de Datos para éstos propósitos, pero la versión final empleó archivos XML, ambas son descritas.

3.1 El Recurso de Internet

A estas alturas resulta difícil no haber sido arrastrado por la ola de Internet, no haber recibido desde un medio u otro, una avalancha de información[12]. Internet puede ser tan simple o compleja como se desee. Al margen de libros, manuales, reportajes, y artículos, genera su propia documentación que circula en los más variados formatos electrónicos y que además muta continuamente y sin previo aviso.

3.1.1 Historia de Internet

Internet surge en septiembre de 1969, cuando la DARPA(*Defense Advanced Research Project Agency*) desarrollo un sistema de transmisión de mensajes eficaz ante un posible ataque nuclear que pudiera destruir una parte de la red de comunicaciones utilizada por el ejercito. La idea no era tanto proteger al máximo el sistema sino ofrecer rutas alternativas que garantizaran la recepción del mensaje a pesar de una destrucción parcial de la red. De este modo se unieron cuatro computadoras mediante un protocolo llamado NCP (*Network Communications Protocol*) formando la red llamada ARPANet. En poco tiempo este protocolo evolucionó hasta el que existe actualmente en Internet, el popular TCP/IP (*Transmission Control Protocol / Internet Protocol*), y de una aplicación puramente militar se pasó a otra más pacífica y constructiva, ya que a dicha red se unieron con el propósito de mantenerse en contacto, de una forma rápida y cómoda: científicos, universidades y centros de investigación de EEUU, con el propósito de poner en común información y hacer participes de sus descubrimientos al resto del mundo tan solo sentándose frente al monitor de su computadora. En poco tiempo (2 años) la DARPA y su red, basada ya en el protocolo TCP/IP, consiguió llegar al 90% de los departamentos de Ingeniería y computadoras de universidades norteamericanas.

El crecimiento tan espectacular que se ha producido en Internet, ha sido en gran medida a la creación de un sistema capaz de incorporar imágenes, gráficos y sonido en las transmisiones, y no solo caracteres como hasta entonces: el World Wide Web(Telaraña de cobertura mundial). La incorporación de este método , ha permitido la entrada en Internet de aplicaciones y servidores más comerciales, y por lo tanto un crecimiento en el número de usuarios domésticos de todo el mundo.

3.1.2 Servicios de Internet

Internet ofrece información y posibilidades de comunicación a través de lo que se denomina *Servicios de Internet.* Estos servicios pueden transmitir mensajes, archivos, multimedia, en pocas palabras una gran variedad de información.A continuación los describimos.

e-mail: Sistema de correo electrónico que nos permite enviar mensajes, gráficos, etc. a cualquier parte del mundo.

listservers: Un mensaje enviado de forma unidireccional, llega a través de

e-mail a miles de ordenadores abonados a esa lista.

FTP: Sistema que nos permite recuperar y grabar archivos en computadoras situadas en cualquier parte del mundo.

chat's: Sistema para hablar simultáneamente (mediante texto) con varias personas a la vez.

news: Mensajes con un remitente pero sin ningún destinatario concreto, que se difunden a través de todas las máquinas conectadas a la red. Actualmente existen más de 10.000 news sobre temas concretos.

WWW: Es sin duda la herramienta más potente e innovadora de Internet. Además de las características antes descritas: transmisión de texto, gráficos, sonido y animaciones, podemos decir que se trata de un sistema de hipertexto a nivel mundial, ya que tan solo haciendo *clic* con el ratón sobre un texto o gráfico situado en la pantalla del ordenador, podemos acceder a información situada en cualquier servidor del mundo. De esta forma una página WEB presentada en pantalla puede contener , por ejemplo, texto procedente de un servidor en México y un dibujo grabado en la computadora central de la universidad de California, Los Angeles,(UCLA).

El WWW(comúnmente conocido como WEB) utiliza el modelo cliente-servidor. Un servidor WEB es un programa que sirve documentos en lenguaje específico de WEB, actualmente el más usado es el HTML(*HiperText Markup Language*), aunque existen otros de nueva creación que por su potencia e innovación prometen sustituir al clásico HTML. Y un cliente, por su parte, es el programa que interactúa con el usuario, pide documentos al servidor y los interpreta para presentarlos en pantalla. Los clientes más utilizados actualmente son Netscape Navigator y Microsoft Explorer.

3.1.3 HTML

Los tres componentes básicos del WEB son el protocolo HTTP, el esquema de direcciones URL y el lenguaje HTML. Hasta que no existieron estos tres elementos no fue posible la construcción del primer servidor y del primer navegador, y por sí solos ya permiten el establecimiento de una comunicación de suficiente calidad. Con el tiempo, se han ido incorporando componentes y protocolos a esta triada, pero el sistema original permanece en esencia. La transferencia sigue realizandose con los comandos de HTTP, la identificación de objetos en la red sigue siendo tarea de URL y los

contenidos, por complejos y dinámicos que sean, se basan en HTML.

A pesar de la rápida evolución del WEB, HTML continúa siendo uno de sus componentes fundamentales que determina a buena parte de los demás: los navegadores se siguen clasificando dependiendo de la versión HTML que interpreten, se trata de integrar JavaScript y los demás elementos dinámicos en la especificación de HTML, y el propio Java necesita de él para tener un soporte visual y un medio de recibir parámetros.

Aunque no lo pueda parecer a simple vista, todos los programas de tratamiento de texto utilizan alguna convención para describir sus documentos y señalar en qué momento hay que mostrar una cabecera, cómo alinear los párrafos, cómo dejar sitio para una figura, entre otros. Lo que ocurre es que en la mayoría de los casos no somos conscientes del uso de este lenguaje porque el programa lo *recubre* y hace innecesario su conocimiento. Esto mismo ocurre cuando un browser visualiza un documento HTML, el usuario sólo ve el texto formateado y los gráficos y enlaces, no ve los comandos que dicen al programa visualizador qué debe mostrar y cómo.

El código HTML consiste en marcas, etiquetas o tags agregadas dentro de un texto normal ASCII. Éstas son identificadas por el protocolo y por tanto por el navegador y permiten el correcto despliegue de los documentos. Cada parte del documento, así como sus características especiales tales como formatos de texto, indicaciones de saltos de línea, nuevos párrafos, tablas, listas numeradas, colores, hiperligas, y demás son definidos mediante una etiqueta. Todas las etiquetas se delimitan por un juego de pico paréntesis `<ETIQUETA>`, la mayoría de los formatos se aplican por secciones.

3.1.4 JavaScript

En Diciembre de 1995, Netscape y Sun anunciaron el surgimiento de Javascript. Su idea principal era la de permitir la generación de guiones(*scripts*) de servidor y de cliente de una manera sencilla. JavaScript es un lenguaje de guiones, interpretado, que puede programarse estructurado y orientado a objetos. Aunque tiene menos capacidad que un lenguaje como C, C++ o Java, JavaScript es lo suficientemente potente para los propósitos orientados a WEB[13]. JavaScript es un lenguaje limitado. Por ejemplo, en JavaScript no se pueden escribir aplicaciones autónomas, y tiene una capacidad muy limitada para leer y escribir archivos. Es mas, los guiones de JavaScript no se pueden ejecutar sin la presencia de un interprete, que puede ser un servidor WEB o un navegador.

Para poder comprender mejor a JavaScript podemos mencionar lo que se puede y no se puede lograr con él:

JavaScript puede:

- hacer reaccionar a las páginas WEB ante eventos del cliente sin recurrir a llamados de servidor.(comúnmente conocido como HTML dinámico);
- preprocesar información en el lado del cliente antes de enviarla al servidor;
- manejar pequeñas cantidades de información de una base de datos en el lado del servidor.

JavaScript no puede:

- Leer archivos en el cliente;
- Escribir archivos en el cliente;
- Leer archivos en el servidor;
- Escribir archivos en el servidor.

3.2 Programación Orientada a Objetos

Actualmente una de las áreas más explotadas en la industria de software y en el ámbito académico es la orientación a objetos. La orientación a objetos promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

3.2.1 ¿Qué es la Programación orientada a objetos?

La *programación orientada a objetos* (POO) es una forma de enfocar la tarea de la programación. Los enfoques de la programación han cambiado drásticamente desde la invención de las computadoras, para acomodarse a la creciente complejidad de los programas. Por ejemplo, cuando se inventaron

las computadoras, la programación se realizaba introduciendo mediante una consola las instrucciones máquina en binario. Esto funcionaba porque los programas sólo tenían unos pocos cientos de instrucciones. Cuando crecieron los programas, se inventó el lenguaje ensamblador para que el programador pudiera manejar programas más largos y complejos usando una representación simbólica de las instrucciones máquina. Los lenguajes de alto nivel aparecieron para proporcionar al programador más herramientas con las cuales gestionar esa complejidad. El primer lenguaje ampliamente utilizado fue FORTRAN[2].

En los años sesenta nace la programación estructurada. Este es el método alentado por lenguajes como C y Pascal. Con los lenguajes estructurados es posible escribir programas moderadamente complejos de una forma bastante sencilla. Sin embargo, cuando los proyectos alcanzan un cierto tamaño, se vuelve demasiado complejos para ser controlados por un programador.

Hoy en día, hay muchos proyectos que están próximos o en el punto donde la aproximación de la programación estructurada ya no funciona. Para resolver este problema se desarrolló la programación orientada a objetos. La programación orientada a objetos toma las mejores ideas de la programación estructurada y las combina con nuevos y poderosos conceptos que animan una nueva visión de la tarea de la programación. La programación orientada a objetos permite descomponer fácilmente un problema en subgrupos de partes relacionadas. Entonces, puede traducir estos subgrupos en unidades auto contenidas llamadas objetos. Todos los lenguajes de programación orientada a objetos tienen siempre tres cosas en común: encapsulación, polimorfismo y herencia. A continuación consideraremos uno a uno éstos conceptos.

Encapsulación

La Encapsulación es el mecanismo que enlaza el código y los datos, al tiempo que protege a ambos frente a interferencias o fallos exteriores. Además, la encapsulación permite creación de un objeto. Un objeto es simplemente una entidad lógica que contiene y código que manipula esos datos. Dentro de un objeto, parte del código o datos son privados del objeto e inaccesibles desde fuera de él. De esta forma, un objeto proporciona un significativo nivel de protección contra modificaciones accidentales contra un uso incorrecto. Un objeto es una variable de un tipo definido por el usuario. Al principio puede parecer extraño pensar en un objeto, que une código y datos, como en una variable. Sin embargo, este es precisamente el caso en

la programación orientada a objetos. Cuando se define un objeto se está creando implícitamente un nuevo tipo de dato.

Polimorfismo

Los lenguajes de programación orientada a objetos admiten el polimorfismo, caracterizado por la frase *una interfaz, múltiples métodos*. En términos más sencillos, el polimorfismo es el método que permite que una interfaz sea utilizada para varios propósitos relacionados pero ligeramente diferentes. La acción específica que se utiliza está determinada por la naturaleza exacta de la situación. El polimorfismo reduce la complejidad permitiendo que la misma interfaz sea utilizada para especificar una clase general de acciones. El compilador será el encargado de determinar la acción específica(es decir, el método) que se aplicará a cada situación. El programador no necesita realizar esta selección de forma manual. Lo único que deberá recordar y utilizar es la interfaz general.

Herencia

La herencia es el proceso por el cual un objeto puede adquirir las propiedades de otro objeto. Esto es importante porque permite manejar el concepto de clasificación . Si se piensa detenidamente, la mayoría del conocimiento se hace manejable por medio de clasificaciones jerárquicas. Sin el uso de clasificaciones habría que definir todas las características de cada objeto explícitamente. Usando clasificaciones, sólo es necesario definir las cualidades que hacen único a un objeto dentro de su clase. La herencia es el mecanismo que hace posible que un objeto sea un ejemplar específico dentro de una clase más general. La herencia es, por tanto, un aspecto importante de la programación orientada a objetos[2].

3.2.2 Java

Java es un lenguaje de programación orientado a objetos, que fue introducido por Sun Microsystems en 1995, y diseñado en principio para el ambiente distribuido de Internet[14].

Entre las cualidades de Java podemos mencionar las siguientes:

Universal

Se han escrito intérpretes de pequeño tamaño adaptados a prácticamente cualquier plataforma, desde mainframes y computadoras personales (con cualquier sistema operativo) hasta dispositivos electrónicos de bajo costo. La mayor parte de los navegadores(Netscape Navigator, Internet Explorer) integran intérpretes de Java, lo que hace posible acceder automáticamente a los applets(pequeñas aplicaciones de Java) presentes en las páginas WEB. También se suele hacer referencia a la universalidad de Java con términos equivalentes como portabilidad, o independencia de plataforma, pues para ejecutar un programa basta compilarlo una sola vez: a partir de entonces, se puede hacer correr en cualquier máquina que tenga implementado un intérprete de Java. Además, las bibliotecas estándar de funciones y métodos de Java facilitan la programación de multitud de acciones complejas (desarrollo de interfaces gráficas, multimedia, multitarea, interacción con bases de datos entre otros). Ningún otro lenguaje dispone como Java de una cantidad tan grande de funciones accesibles en cualquier plataforma sin necesidad de cambiar el código fuente.

Sencillo

Java es un lenguaje de gran facilidad de aprendizaje, pues en su concepción se eliminaron todos aquellos elementos que no se consideraron absolutamente necesarios. Uno de los aspectos más notables de Java con respecto a lenguajes como C o C++ es la ausencia de apuntadores, o lo que es lo mismo: es imposible hacer referencia de forma explícita a una posición de memoria; esto ahorra gran cantidad de tiempo a los programadores, dado que el comportamiento imprevisto de los apuntadores es una de las principales fuentes de errores en la ejecución de un programa. Por otro lado, Java dispone de un mecanismo conocido como de *recogido de basura*, el cual hace que, durante la ejecución de un programa, los objetos que ya no se utilizan se eliminen automáticamente de la memoria. Dicho mecanismo facilita enormemente el diseño de un programa y optimiza los recursos de la

máquina que se esté usando para la ejecución del mismo (con los lenguajes tradicionales, la eliminación de objetos y la consiguiente optimización de recursos debe planificarse cuidadosamente al idear el programa).

Orientado a Objetos

Java es un lenguaje orientado a objetos desde su concepción. Maneja los conceptos y ventajas del Encapsulamiento de Datos, Herencia, Polimorfismo que fueron descritos con anterioridad.

Seguro

En general, se considera que un lenguaje es tanto más seguro cuanto menor es la posibilidad de que errores en la programación, o diseños malintencionados de programas (virus), causen daños en el sistema.

Java tiene una ausencia de manejo de apuntadores (que evita cualquier error de asignación de memoria). También tiene el *ocultamiento de la información* propio de la programación orientada a objetos. Las aplicaciones que viajan en Internet son manejadas en un entorno conocido como caja de arena(*sandbox*), que no permite que aplicaciones mal intencionadas afecten el entorno en el que son descargadas. El diseño de los métodos y clases de las bibliotecas estándar de Java, realizan múltiples verificaciones cuando son invocados, de modo que se dificultan los errores ya sean voluntarios o involuntarios.

3.2.3 JSP

Java cuenta con dos propuestas para efectuar aplicaciones sobre servidores WEB. Éstas son los Java Servlets y JSP(*Java Server Pages*). Notamos que JSP, es una extensión de la tecnología de Java Servlets.

Las tecnologías antes mencionadas, son empleadas para implementar entre otras cosas:

- Típicos sistemas con interacción a Bases de Datos en WEB
- Automatización de sistemas de recepción y publicación de información en WEB
- Control de recepción de e-mails, y de sistemas de news, chats, entre otros. Conviene recordar que Java está especialmente indicado para la programación utilizando protocolos TCP/IP

- Pueden generar páginas HTML en el cliente de forma dinámica
- Dado que pueden manejar múltiples peticiones concurrentemente, es posible implementar aplicaciones colaborativas

Como parte de la familia de Java, la tecnología de JSP permite el desarrollo rápido de las aplicaciones basadas en WEB que son independientes de plataforma. Además, en particular la tecnología de JSP permite separar la interfaz del usuario de la parte lógica del proyecto. Esto permite a los diseñadores cambiar la apariencia de las páginas totalmente, sin alterar el contenido dinámico subyacente. Lo anterior es logrado insertando código de Java dentro de las páginas WEB(implementadas en HTML), que se encarga de generar el contenido dinámico.

3.2.4 JavaBeans

Los JavaBeans son definidos como *componentes de software reuables que se pueden manipular visualmente en una herramienta de construcción*[15].

Todo objeto que se ajuste a ciertas reglas básicas de diseño puede ser un bean; no existe una clase Bean que todos los beans necesiten subclasificar. Muchos beans son componentes gráficos; pero también resulta totalmente válido y útil escribirlos *invisibles*, sin una apariencia en pantalla.(Sin embargo, aunque no tenga apariencia en la pantalla dentro de una aplicación terminada, es posible manipularlo mediante herramientas visuales).

Un bean exporta propiedades,eventos y métodos. Una *propiedad* es un fragmento del estado interno del bean, que se puede establecer y consultar programándolo, por lo general mediante un par estándar de métodos de acceso `get` y `set`. El acceso directo a las propiedades es conocido como introspección. Un bean puede generar *eventos* así como lo hacen los componentes gráficos, como botones(los que reaccionan ante los eventos de ser oprimido, ser enfocado, etcétera). Finalmente, los *métodos* exportados por un bean son los métodos `public` definidos por él mismo, excepto aquellos métodos empleados para obtener y establecer valores de propiedad y para registrar y retirar oyentes de eventos.

En particular, las clases que interactúan con los JSP's son pensados como beans. Esto debido a que son reusados por los JSP's. Además, los JSP's al incluir clases(que cumplen con el patrón de diseño para las propiedades) les definen la *introspección*, que permite alterar y obtener los valores de las propiedades privadas de manera directa(es decir, sin el uso de métodos).

3.3 Bases de Datos

El almacenamiento y gestión de la información es un aspecto importante del desarrollo de software. Anteriormente la información era almacenada como archivos sin ninguna estructura, generando muchos problemas como redundancia, inconsistencia, dificultad de acceso, por mencionar algunos. Con el tiempo, se generaron mecanismos para evitar los problemas antes mencionados y garantizar un correcto almacenamiento y gestión de la información.

Podemos pensar a una base de datos como un repositorio de datos almacenados, y, en general, es tanto *integrada* como *compartida*[17].
Por *integrada* se entiende que la base de datos puede considerarse como una unificación de varios archivos de datos independientes, donde se elimina parcial o totalmente cualquier redundancia entre los mismos.
Por *compartida* se entiende que partes individuales de la base de datos pueden compartirse entre varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos(y utilizarla con propósitos diferentes). Tal comportamiento es consecuencia del hecho de que la base de datos es integrada.

Los mecanismos que se encargan de la gestión de la información contenida en una base de datos son conocidos como Sistemas Manejadores de Bases de Datos (SMBD).

Para poder generar una base de datos es necesario efectuar un diseño de la misma. Un modelo de diseño de bases de datos es *una colección de herramientas conceptuales para describir los datos, las relaciones de datos, la semántica de los datos y sus reglas de consistencia*[16]. Hay varias modelos orientados al diseño de una base de datos, entre los más usados tenemos al Modelo Entidad-Relación,y el Modelo Relacional.

3.3.1 Modelo Entidad-Relación

Propuesto por Peter Chen en 1976, éste modelo es una representación gráfica para el modelado de las bases de datos. Se desarrollo para facilitar el desarrollo de bases de datos permitiendo la especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos.

El modelo consta de tres componentes principales:

Entidades: Conjuntos de cosas u objetos en el mundo real que son distinguibles de otros conjuntos de objetos

Atributos: Son valores que describen las características de una entidad

Relaciones: Es una asociación ente varias entidades

3.3.2 Modelo Relacional

El modelo relacional fue propuesto originariamente por Tedd Codd en un ya famoso artículo de 1970. Gracias a su coherencia y facilidad de uso, el modelo se ha convertido en los años 80 en el más usado para la producción de SMBD. Incluso muchos de los diseños generados en el modelo que se haya elegido, al ser llevados a su implementación son convertidos a éste modelo. Esto se debe en gran parte a que SQL(*Structured Query Language*), que es el lenguaje más importante de peticiones, está basado en éste modelo.

La estructura fundamental del modelo relacional es una tabla bidimensional constituida por renglones(tuplas) y columnas(atributos) llamada *relación*. Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada ejemplar de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad.

3.4 XML

XML está basado en el Lenguaje Estándar Generalizado de Marcas(*Standar Generalized Markup Language* o *SGML*), que fué definido en 1986 por la Organización Internacional de Standarización(*International Standar Organization o ISO*) como el stándar ISO 8879.

XML es un formato que hace que la información contenida en un documento sea auto-descriptiva. La norma para el XML está controlada por el Consorcio Mundial de Web (World Wide Web Consortium, o W3C). A continuacuón se presenta la descripción formal del XML según su sitio WEB en `http://www.w3.org/TR/1998/REC-xml-19980210`

> "El Lenguaje Extensible de Marcado o Extensible Markup Language, abreviado XML, describe una clase de objetos de datos llamados documentos XML y describe parcialmente el comportamiento de los programas de ordenador que los procesan... Los documentos XML están formados por unidades de almacenamiento llamadas entidades, las cuales contienen datos analizados sintácticamente o no (parsed o unparsed).

> Los datos analizados están formados por caracteres, algunos de los cuales forman datos de carácter, mientras que otros forman el marcado (markup). El Marcado codifica una descripción de la disposición y la estructura lógica del almacenamiento del documento. El XML provee un mecanismo para imponer limitaciones sobre la disposición del almacenamiento y su estructura lógica"

3.4.1 Ventajas y Desventajas de XML

XML hace fácil el intercambiar información entre aplicaciones de una misma organización o de distintas organizaciones. XML es también útil para integrar diversas fuentes de información, y presentar una interfaz uniforme de éstas. De cualquier manera XML no es una solución general para todos los problema. Como todas las tecnologías, tiene sus propias ventajas y desventajas.

Ventajas:

- Extensible: Se pueden añadir elmentos personales a los documentos XML
- Interoperable: XML no depende del sistema operativo, lenguaje o fuente de información de la aplicación que los usa
- Datos Auto-Descriptivos: La intención con XML es que los datos sean auto-descriptivos, de tal forma que su estructura sea fácilmente identificada en aplicaiones futuras

Desventajas:

- Estándar: XML es una teconología relativamente inmadura, y el estándar de XML está aún definiendose en ciertas áreas. Ésta situación se mejorará con el paso de los años, conforme los estándares de XML sean completamente establecidos. Mientras tanto, las organizaciones deben tomar decisiones estrategicas sobre los aspectos de XML que van a usar.

Capítulo 4

Desarrollo del Sitio WEB TSPi

A continuación se presenta una revisión del desarrollo del sitio de TSPi.En ésta se tratan la descripción del desarollo del sitio, un resumen de las especificaciones finales del sitio y las primeras experiencias obtenidas del uso del mismo. Destacamos el uso del Lenguaje Unificado de Modelado, mejor conocido por sus siglas UML(*Unified Model Language*), en algunas partes de éste capítulo.

4.1 Descripción de Desarrollo

Para llevar a cabo el desarrollo del proyecto usando la metodología de TSPi, desempeñé cada uno de los distintos roles que la metodología propone. Así también, desempeñé el papel de desarrollador y diseñador del sitio.

Básicamente se llevaron los siguientes componentes al sitio WEB:

- Actividades TSPi, Ciclo 1 y Ciclo n: Diagramas, Actividades, Productos, Ejemplos y Guiones.
- Fase de Lanzamiento del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos.
- Fase de Estrategia del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos.
- Fase de Planeación del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos.
- Fase de Requerimientos del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos

- Fase de Diseño del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos
- Fase de Implementación del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos.
- Fase de Pruebas del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos.
- Fase de Postmortem del Ciclo 1 y n: Diagramas, Actividades , Productos, Ejemplos.
- Explicación sobre TSPi y Explicación de Diagramas del Sitio.

Para completar el proyecto, se llevaron a cabo cuatro ciclos de desarrollo de la metodología de TSPi. A continuación doy una breve explicación de las metas de dichos ciclos, si se quiere consultar la documentación generada en cada ciclo de desarrollo lea el Apéndice A.

4.1.1 Ciclo 1 de Desarrollo

En este ciclo las partes del sitio sobre las que se trabajaron fueron: Descripción TSPi, Ciclo 1 y Ciclo n: Diagramas, Actividades, Productos, Ejemplos y Guiones. Fase de Lanzamiento del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos. Fase de Estrategia del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos.

Debido a que éste ciclo es el primero, se tuvieron que tomar decisiones sobre los formatos de los diagramas, ejemplos, actividades, y productos de la metodología y sobre el diseño de la interfaz gráfica del sitio.

Para poder efectuar un proyecto inicial muy acercado a las necesidades del cliente y de acuerdo a la metodología de TSPi; se llevó a cabo (entre otros) la elaboración del documento de Esclarecimiento de Necesidades, en el que se encuestó a los clientes del proyecto, a compañeros del curso de Ingeniería de Software (impartido en la Facultad de Ciencias de la UNAM) que habían usado la metodología TSPi , así como a compañeros que llevarían el mismo curso y aprenderían la metodología. A partir de los resultados obtenidos se tomaron las primeras decisiones, las cuáles fueron modificados en repetidas ocasiones para obtener siempre una mejor propuesta. Las modificaciones que posteriormente se realizaron eran en función de las observaciones realizadas por los clientes, por colegas que conocían la metodología y por expertos en el diseño de interfaces.

Fueron documentadas todas las fases de TSPi para éste ciclo y los documentos obtenidos se archivaron como parte del proyecto.

Las herramientas usadas en éste ciclo fueron HTML y programación en JavaScript para dar vista al proyecto en el browser. En particular se recurrió al editor visual ©DreamWeaver. Se usó ©Photoshop, ©Corel Draw, y ©Acrobat Reader para dar formato a los diagramas y ejemplos del sitio.

4.1.2 Ciclo 2 de Desarrollo

En este ciclo las partes del sitio sobre las que se trabajaron fueron: Fase de Planeación del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos. Fase de Requerimientos del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos. Fase de Diseño del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos.

En este ciclo se determinó definitivamente el formato y estrategia de generación de los diagramas, ejemplos, actividades y guiones usados en el sitio.

En este punto se decidió que el sitio se llevaría a una versión en JSP para que fuera flexible en su mantenimiento. Esto se realizaría una vez generado todo el material que contendría el sitio.

Fueron documentadas todas las fases de TSPi para éste ciclo y los documentos obtenidos se archivaron como parte del proyecto.

En este caso se continuó el uso de las herramientas empleadas en el ciclo anterior.

4.1.3 Ciclo 3 de Desarrollo

En este ciclo las partes del sitio sobre las que se trabajaron fueron: Fase de Implementación del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos. Fase de Pruebas del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos. Fase de Postmortem del ciclo 1 y n, con sus diagramas, actividades, productos, y ejemplos. Correcciones de Fases Anteriores

Para este ciclo ya se tenía bien definido el trabajo a hacer y éste sólo consistía en una continuación lógica del ciclo anterior. Sin embargo, en éste ciclo se me entregaron modificaciones a algunos de los materiales ya generados por efectos de mayor claridad en la abstracción de los diagramas UML sobre la metodología TSPi.

Fueron documentadas todas las fases de TSPi para éste ciclo y los documentos obtenidos se archivaron como parte del proyecto.

Las herramientas empleadas fueron las mismas que en los ciclos anteriores.

4.1.4 Ciclo 4 de Desarrollo

En este ciclo se llevó el sitio WEB , que era estático, a una versión que usa Java Server Pages (JSP), básicamente esto responde a la necesidad de tener una manera más fácil de realizar algún cambio o actualización del sitio. Para poder realizar ésta modificación se recurrió al uso de compiladores de Servlets, particularmente de JSPs como lo es la distribución de Jakarta Tomcat 4.0.1 y tener en cuenta sus propiedades.

Para este ciclo, se llevó a cabo un cambio en los productos obtenidos al término del mismo. Particularmente los productos alterados fueron el diseño de alto nivel, el diseño detallado y código.

El cambio consistió en la sustitución de una base de datos empleada por los JSPs, y en su lugar se recurrió al manejo de archivos XML. El cambio responde a una solicitud del cliente debido a una posible distribución del sitio en CD. Ésto con la intención de que no sea indispensable tener acceso a la red para poder consultar el sitio.

El Sistema Manejador de Bases de Datos(SMBD) empleado para la primera versión del ciclo 4, era PostgreSQL 7.0.3. El parser XML, que se empleó para la versión final fué SAX.

También en este ciclo se llevaron a cabo algunas otras modificaciones en la interfaz.

Fueron documentadas todas las fases de TSPi para éste ciclo y los documentos obtenidos se archivaron como parte del proyecto. El sitio puede ser actualmente consultado en `http://kasia.fciencias.unam.mx/TSPi`

Las herramientas usadas en éste ciclo fueron programación en Java, HTML, programación en JavaScript, el editor ©DreamWeaver, y de PostgreSQL y archivos XML para la presentación del sitio en el browser.

4.2 Resumen de Especificación de Requerimientos

A continuación se presentan los requerimientos para el desarrollo del sitio. Éstos fueron generados antes de cualquier desarrollo de cada ciclo, con la finalidad de que el producto final cubriera las necesidades del cliente. Por otro

lado, son de gran utilidad para el desarrollo del sistema, porque constituyen un mapa global sobre el cuál se va a trabajar.

4.2.1 Descripción de Necesidades del Sitio Web TSPi

A continuación se mencionan e identifican los componentes que conforman las necesidades del proyecto.

1.1 Generar Introducción del sitio

1.2 Llevar a WEB introducción del sitio en texto

1.3 Llevar a WEB mapas guías de metodología TSPi

2.1 Llevar a WEB Lanzamiento del Ciclo 1

2.2 Llevar a WEB Lanzamiento del Ciclo n

3.1 Llevar a WEB Estrategia del Ciclo 1

3.2 Llevar a WEB Estrategia del Ciclo n

4.1 Llevar a WEB Planeación del Ciclo 1

4.2 Llevar a WEB Planeación del Ciclo n

5.1 Llevar a WEB Requerimientos del Ciclo 1

5.2 Llevar a WEB Requerimientos del Ciclo n

6.1 Llevar a WEB Diseño del Ciclo 1

6.2 Llevar a WEB Diseño del Ciclo n

7.1 Llevar a WEB Implementación del Ciclo 1

7.2 Llevar a WEB Implementación del Ciclo n

8.1 Llevar a WEB Pruebas del Ciclo 1

8.2 Llevar a WEB Pruebas del Ciclo n

9.1 Llevar a WEB Postmortem del Ciclo 1

9.2 Llevar a WEB Postmortem del Ciclo n

4.2.2 Requerimientos Funcionales

En éste punto, se describirán los requerimientos funcionales empleando casos de uso de UML.El propósito de los casos de uso es el de modelar la relación de una aplicación o sistema con su entorno, siendo éste constituido por usuarios u otras aplicaciones.
El diagrama de casos de uso que se muestra en la Figura 4.1 es el **Diagrama General de Casos de Uso** del sitio. El resto de ésta sección está dedicado a detallar las partes del diagrama mencionado anteriormente.

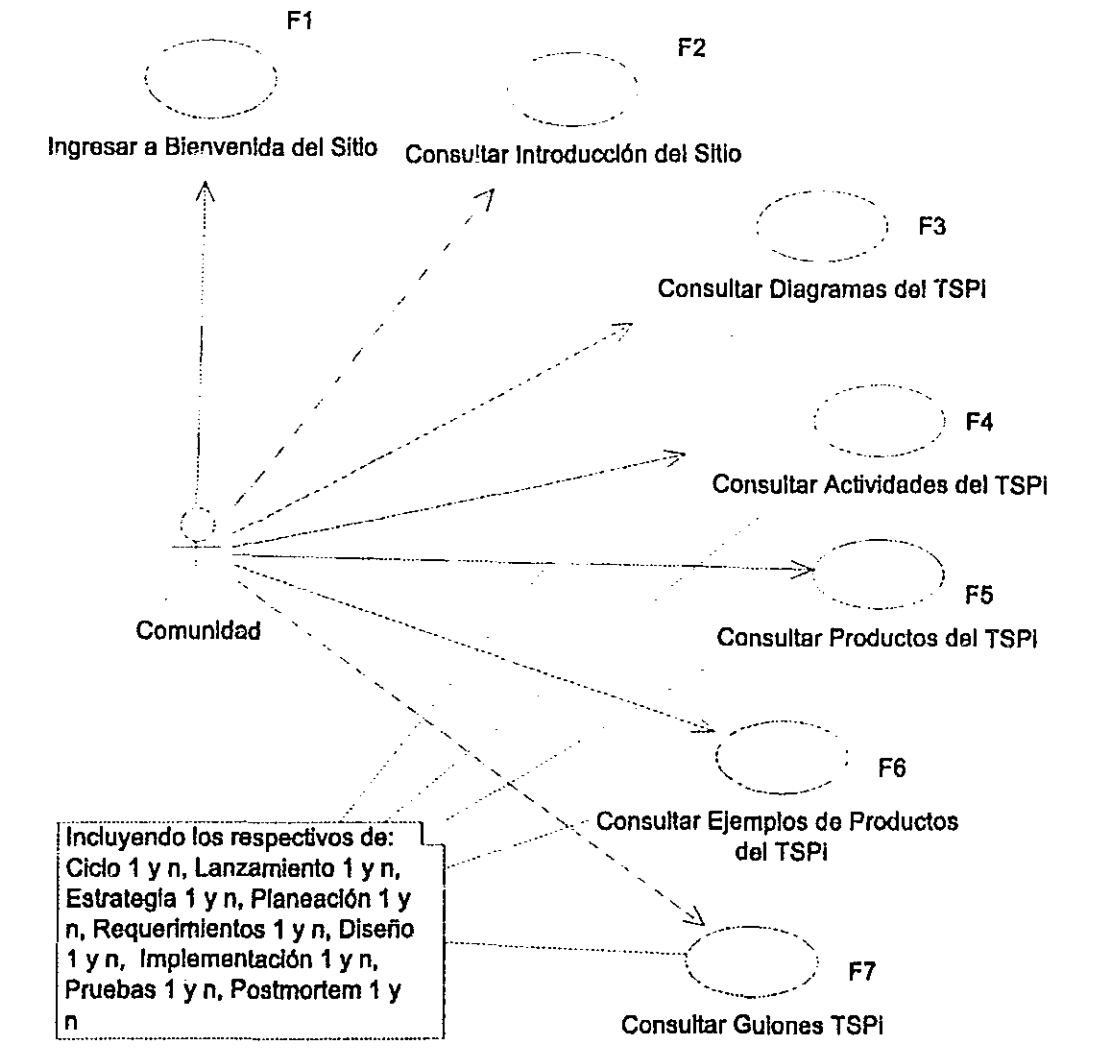


Figura. 4.1: Caso de Uso General del Sitio.

CASO DE USO: Acceder a Bienvenida del Sitio

IDENTIFICADOR: F1

ACTORES: Comunidad

DESCRIPCIÓN: En este punto se muestra la pantalla de bienvenida del sitio. En ésta se describirá a grandes rasgos la intención del sitio y se presentarán las ligas para comenzar la metodología o bien accesar a una introducción del sitio

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia la bienvenida del sitio	2	La página de bienvenida es desplegada	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.1: Flujo de Caso de Uso para F1.

CASO DE USO: Acceder a Página de Introducción

IDENTIFICADOR: F2

ACTORES: Comunidad

DESCRIPCIÓN: En este punto se da una explicación de cuál es la meta de la metodología de TSPi, ésto satisface los componentes 1.1 y 1.2 de la Descripción de Necesidades. Aquí notamos la existencia de las ligas de bienvenida al sitio y de TSPi

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia la introducción del sitio	2	La página muestra la sección la introducción deseada	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.2: Flujo de Caso de Uso para F2.

CASO DE USO: Acceder a Página de Diagrama(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n)

IDENTIFICADOR: F3

ACTORES: Comunidad

DESCRIPCIÓN: En este punto se muestran diagramas de estados(*statechart*) o de flujo(*workflow*), los cuáles describen la metodología de TSPi. Ésta pantalla satisface los componentes 1.3, 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 8.1, 8.2, 9.1, 9.2 de la Descripción de Necesidades. Notamos que en la pantalla habrán zona sensible y ligas a otras páginas

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia un diagrama TSPi del sitio	2	Se carga una página con el diagrama de la metodología deseado en formato de imagen con características de mapa sensible	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.3: Flujo de Caso de Uso para F2.

CASO DE USO: Acceder a Página de Actividad(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n)

IDENTIFICADOR: F4

ACTORES: Comunidad

DESCRIPCIÓN: En éste tipo de páginas se muestran textos que describen las actividades de la metodología de TSPi. Ésta pantalla satisface los componentes 1.3 , 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 8.1, 8.2, 9.1, 9.2 de la Descripción de Necesidades. Notamos que en la pantalla habrán zona sensible y ligas a otras páginas

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia un actividad del sitio	2	Se carga una página con el texto en HTML de la actividad solicitada	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.4: Flujo de Caso de Uso para F4.

CASO DE USO: Acceder a Página de Producto(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n

IDENTIFICADOR: F5

ACTORES: Comunidad

DESCRIPCIÓN: En éste tipo de páginas se muestran textos que describen los producto de la metodología de TSPi. Ésta pantalla satisface los componentes 1.3, 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 8.1, 8.2, 9.1, 9.2 de la Descripción de Necesidades. Notamos que en la pantalla habrán ligas a otras páginas

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia algún producto. En particular éste es una lista	2	Se carga una página con el texto en HTML de una lista de productos a generar según lo solicitado de productos	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.5: Flujo de Caso de Uso para F5.

CASO DE USO: Acceder a Página de Ejemplo de Producto(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n)

IDENTIFICADOR: F6

ACTORES: Comunidad

DESCRIPCIÓN: En éste tipo de páginas se muestran imagenes que son ejemplos los producto de la metodología de TSPi. Ésta pantalla satisface los componentes 1.3, 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 8.1, 8.2, 9.1, 9.2 de la Descripción de Necesidades. Notamos que en la pantalla habrán ligas a páginas que continúen el ejemplo, ligas a otras páginas, y a downloads

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia algún producto. En particular éste es un ejemplo de producto	2	Se carga una página con el texto en imagen del producto solicitado, y un download del mismo	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.6: Flujo de Caso de Uso para F6.

CASO DE USO: Acceder a Página de Guiones(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n)

IDENTIFICADOR: F7

ACTORES: Comunidad

DESCRIPCIÓN: En éste tipo de páginas se muestran imagenes que describen los guiones de la metodología de TSPi. Ésta pantalla satisface los componentes 1.3, 2.1, 2.2, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 6.1, 6.2, 7.1, 7.2, 8.1, 8.2, 9.1, 9.2 de la Descripción de Necesidades. Notamos que en la pantalla habrán ligas a páginas que continúen el ejemplo, ligas a otras páginas, y a downloads

FLUJO:

ACTOR		SISTEMA		
Paso	Acción	Paso	Acción	Excp
1	El usuario genera la acción de navegar hacia un guión del sitio	2	Se carga una página con el texto en imagen del guión solicitado, y un download del mismo	
		3	El sitio espera una nueva petición por parte del usuario	

Tabla. 4.7: Flujo de Caso de Uso para F7.

4.2.3 Interfaces Externas de los Requerimientos

Las especificaciones del ambiente de desarrollo son muy importantes así como la interfaz con el usuario. En este caso el ambiente de trabajo es Linux y Windows 98. Las salidas son pensadas para Netscape Navigator (desde versión 4 en adelante) e Internet Explorer (desde versión 4 en adelante).

4.2.4 Requerimientos del Diseño e Implementación

El desarrollado del sitio fué pensado con uso de la tecnología JSP. Por lo que se requiere un servidor que soporte el software de Tomcat 4.0.1. También se requiere de la existencia de Java Development Kit en el servidor(1.3.1 o mayor).

El desarrollado del sitio contempla el uso de las herramientas: ©Dream Weaver 3, ©Adobe Photoshop 5.5, y ©Corel Draw 10. Aparte de recurrir a Java Script (al menos versión 1.2).

4.2.5 Requerimientos Especiales del Sitio

El Sitio WEB TSPi toma en cuenta:

Portabilidad: Debido a que el Sitio Web TSPi se implementó en JSP, se garantiza un alto porcentaje de compatibilidad en la ejecución del sistema en diferentes plataformas.

Desempeño: Se encuentra dentro de los límites tolerables, para sitios de consulta de información. Esto es, los tiempos de respuesta en la mayor parte de los casos dependerá del servidor.

Usabilidad: Se respeta en su construcción, elementos estándares de los navegadores en su interfaz gráfica para hacer intuitivo su uso.

Mantenible: Debido a que la construcción del sitio se basa en la documentación, se estima que cualquier modificación se podrá realizar con su usó.

4.3 Resumen de Especificación de Diseño

4.3.1 Diseño Arquitectónico

Para el módulo se maneja una arquitectura común de los sistemas de información por Internet, que abarca una interfaz para el usuario y el almacenamiento persistente de datos, conocida con el nombre de arquitectura de tres capas. Está constituida por las siguientes capas :

- **Capa de Presentación**, en esta capa se ubicarán los elementos con los que interactuará el usuario. Por ejemplo: menús, ventanas, botones, etc.
- **Capa Lógica de la Aplicación**, en esta capa se definen las tareas y reglas que rigen el negocio. Está constituida por los siguientes elementos:
  - Objetos del Dominio del Problema, que son las clases que representan los conceptos del dominio.
  - Servicios, que son aquellos objetos de dominio no relacionados con el problema que prestan el servicio de soporte, como la interacción con la base de datos, los reportes, las comunicaciones, etc.
- **Capa de Almacenamiento**, en esta capa se encuentran los mecanismos de almacenamiento persistente.

4.3.2 Diagrama de Paquetes

El diagrama de paquetes denota un mecanismo para organizar elementos del modelo, subconjuntos del modelo y diagramas. Los paquetes se pueden anidar dentro de otros paquetes.
En la Figura 4.2 se muestra el **Diagrama de Paquetes** para el presente proyecto.

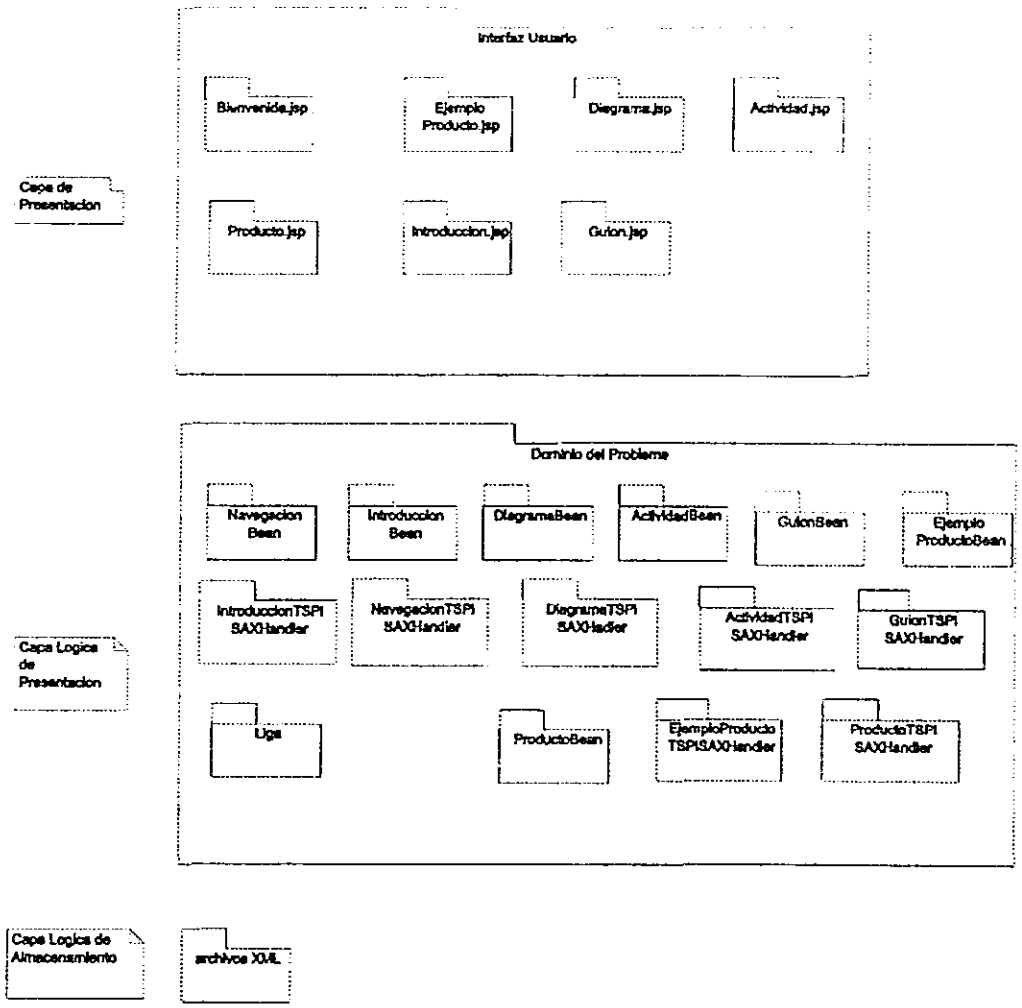


Figura. 4.2: Diagrama de Paquetes.

4.3.3 Diagrama de Clases

Los diagramas de clases describen las entidades en un sistema o dominio y la forma en que tales entidades se relacionan entre sí. En la siguiente página se presenta el **Diagrama de Clases General** para el presente proyecto.

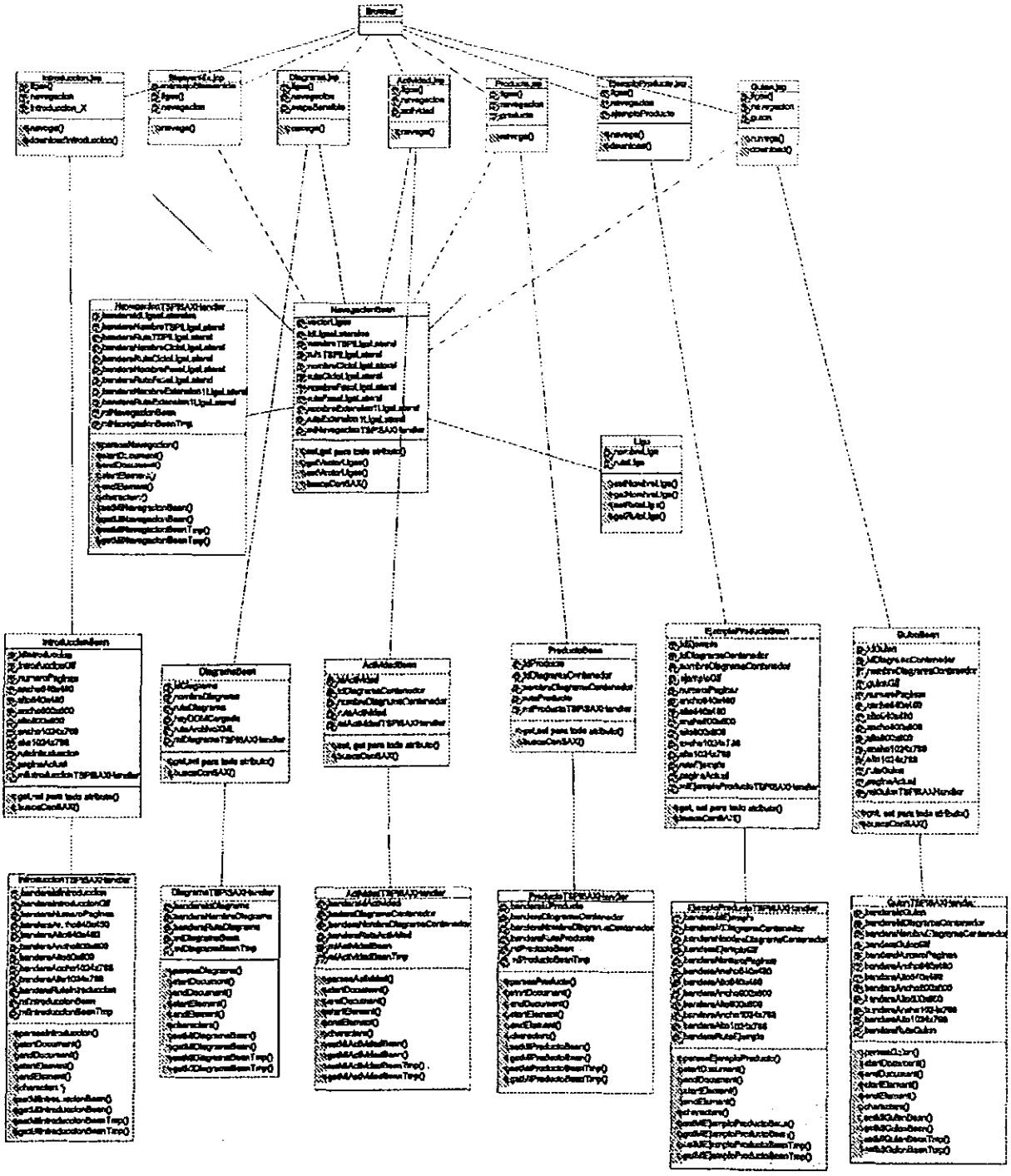


Figura. 4.3: Diagrama de Clases General.

4.3.4 Diagramas de Interacción

Para representar la Interacción del sitio, usaremos los diagramas de secuencia. Los diagramas de secuencia representan la forma en que interaccionan los objetos entre sí al paso del tiempo. Los diagramas de secuencia a su vez, muestran una descripción más detallada de los casos de uso.

Los tipos de pantallas del sitio tienen un funcionamiento similar. En éstas se llevan a cabo consultas sobre los archivos XML para el despliegue de la información adecuada, como lo son los diagramas, productos, actividades o guiones solicitados por el usuario.

CASO DE USO: Acceder a Bienvenida del Sitio

IDENTIFICADOR: F1

DIAGRAMA DE SECUENCIA:

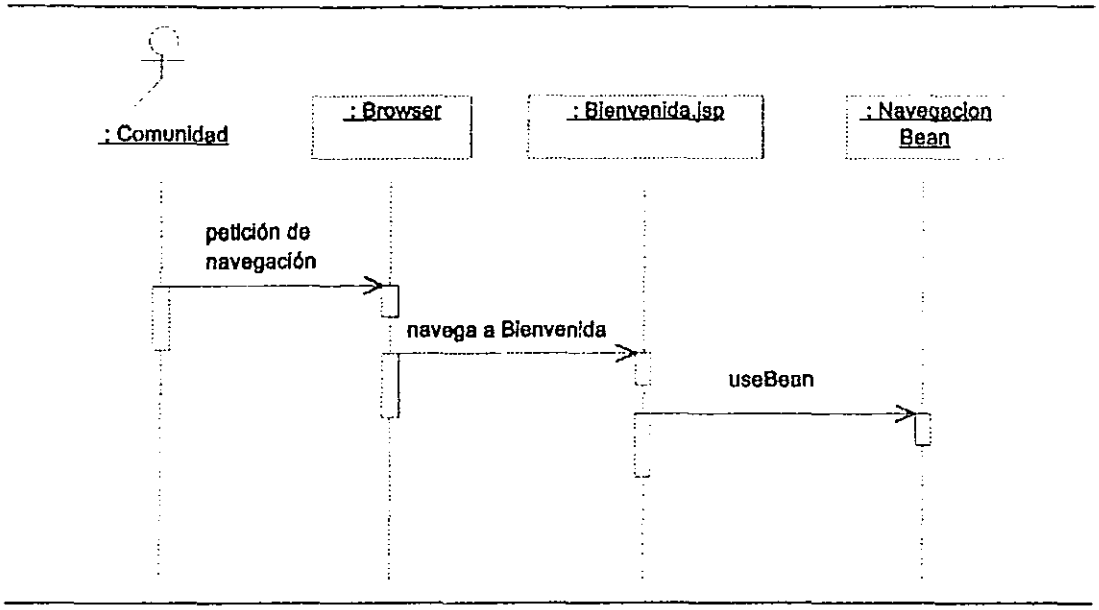


Figura. 4.4: Diagrama de Secuencia para F1.

CASO DE USO: Acceder a Página de Introducción

IDENTIFICADOR: F2

DIAGRAMA DE SECUENCIA:

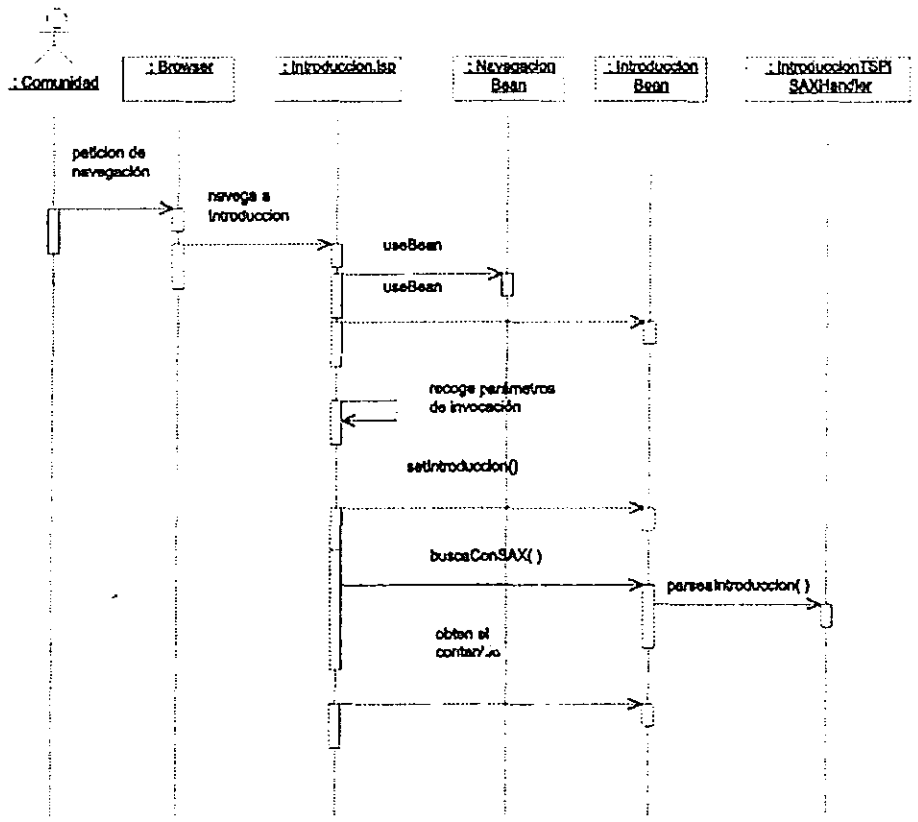


Figura. 4.5: Diagrama de Secuencia para F2.

CASO DE USO:	Acceder a Página de Diagramas(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n, Pruebas 1 y n, Postmortem 1 y n)
IDENTIFICADOR:	F3

DIAGRAMA DE SECUENCIA:

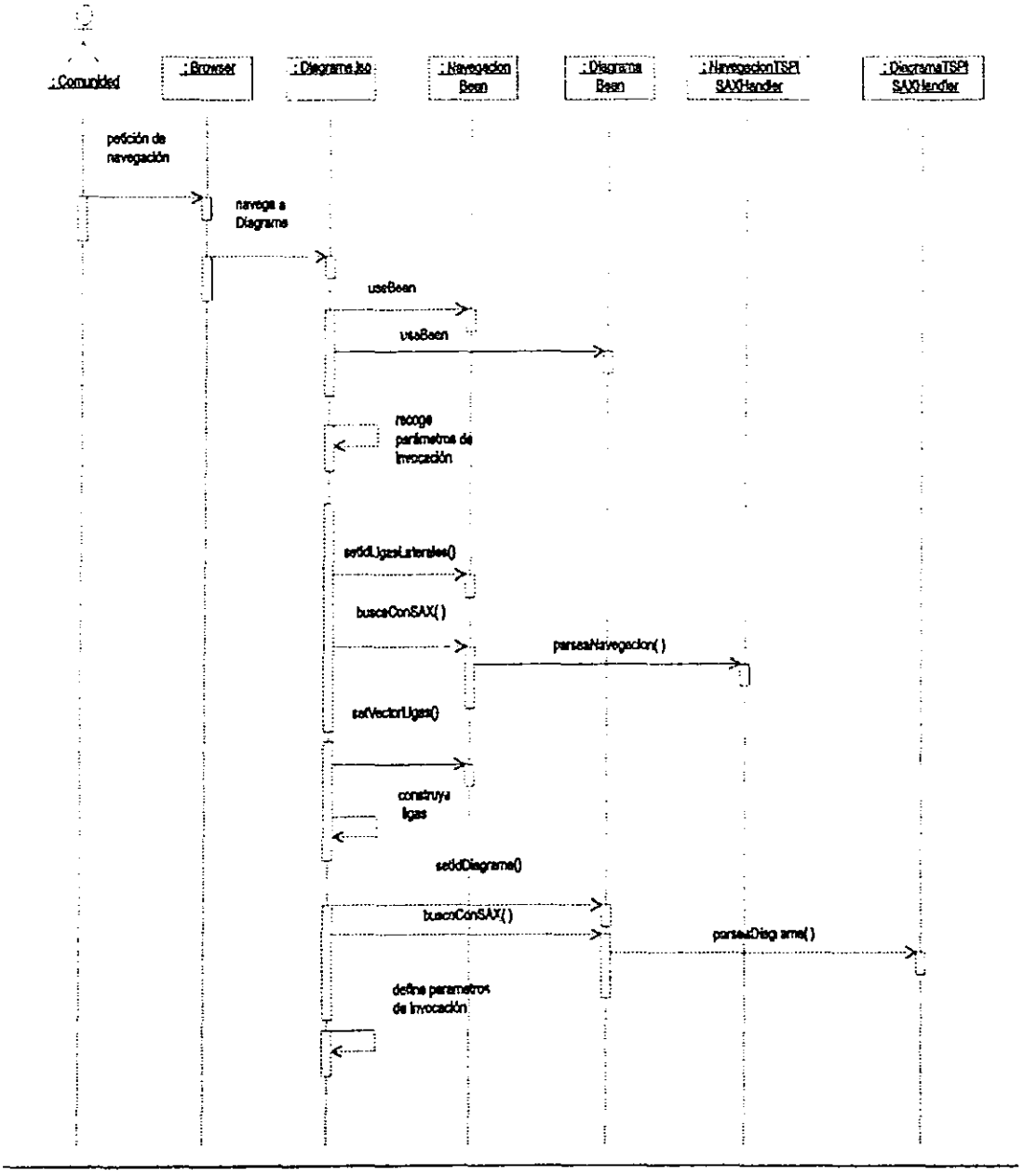


Figura. 4.6: Diagrama de Secuencia para F3.

CASO DE USO:	Acceder a Página de Actividades(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n. Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n, Pruebas 1 y n, Postmortem 1 y n)
IDENTIFICADOR:	F4

DIAGRAMA DE SECUENCIA:

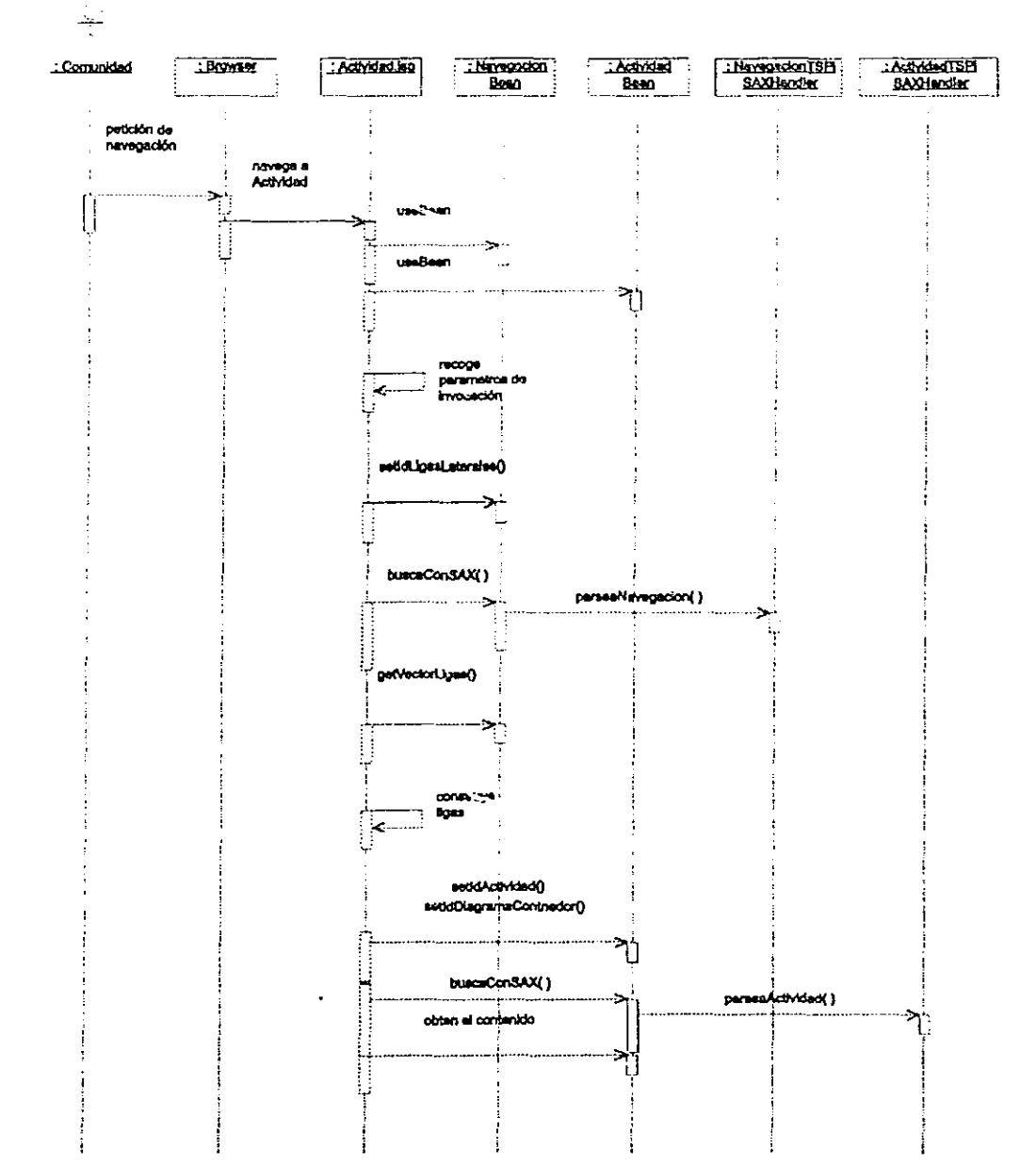


Figura. 4.7: Diagrama de Secuencia para F4.

CASO DE USO:	Acceder a Página de Productos(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n, Pruebas 1 y n, Postmortem 1 y n)
IDENTIFICADOR:	F5

DIAGRAMA DE SECUENCIA:

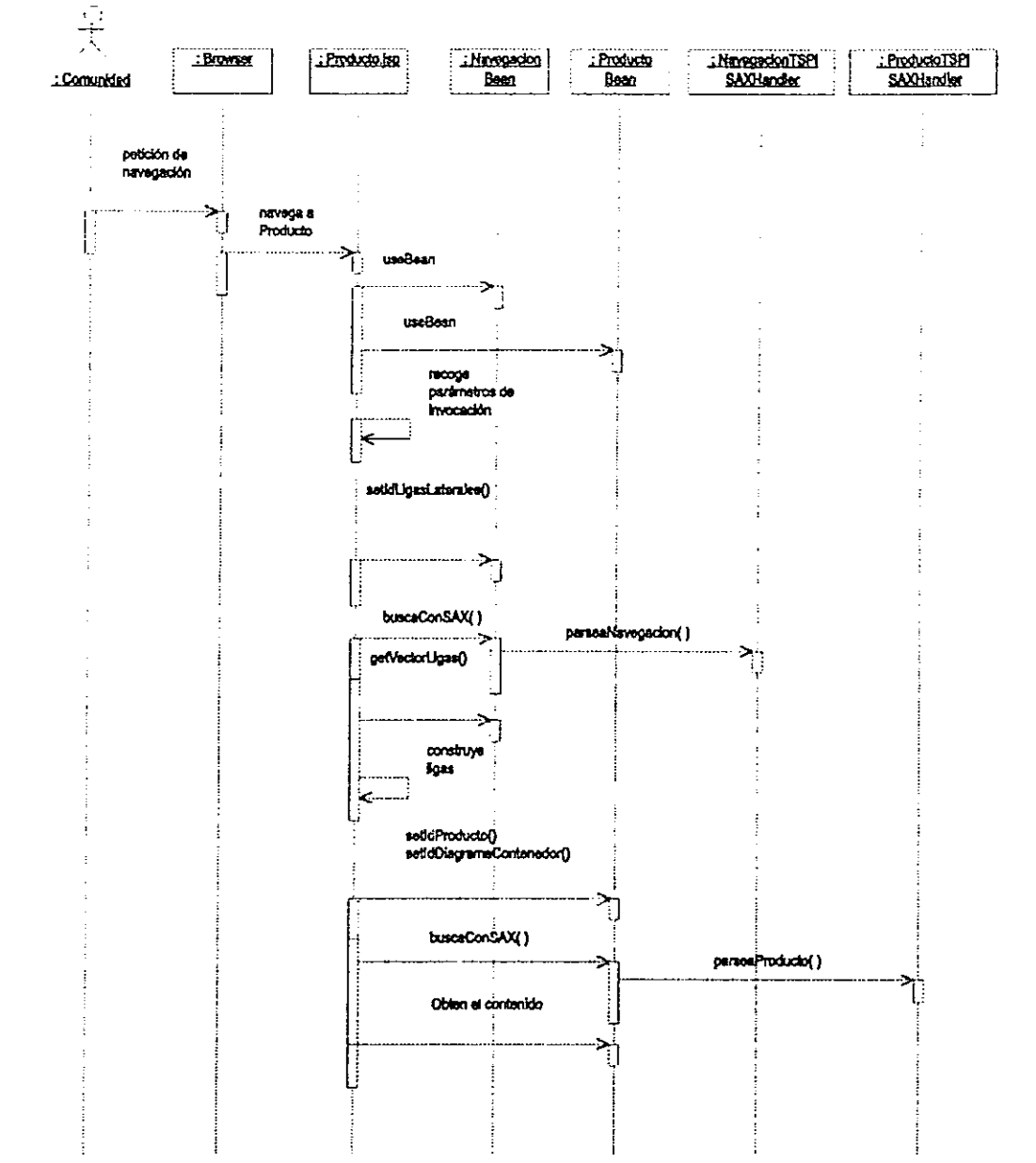


Figura. 4.8: Diagrama de Secuencia para F5.

CASO DE USO:	Acceder a Página de Ejemplos de Productos (para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n, Pruebas 1 y n, Postmortem 1 y n)
IDENTIFICADOR:	F6

DIAGRAMA DE SECUENCIA:

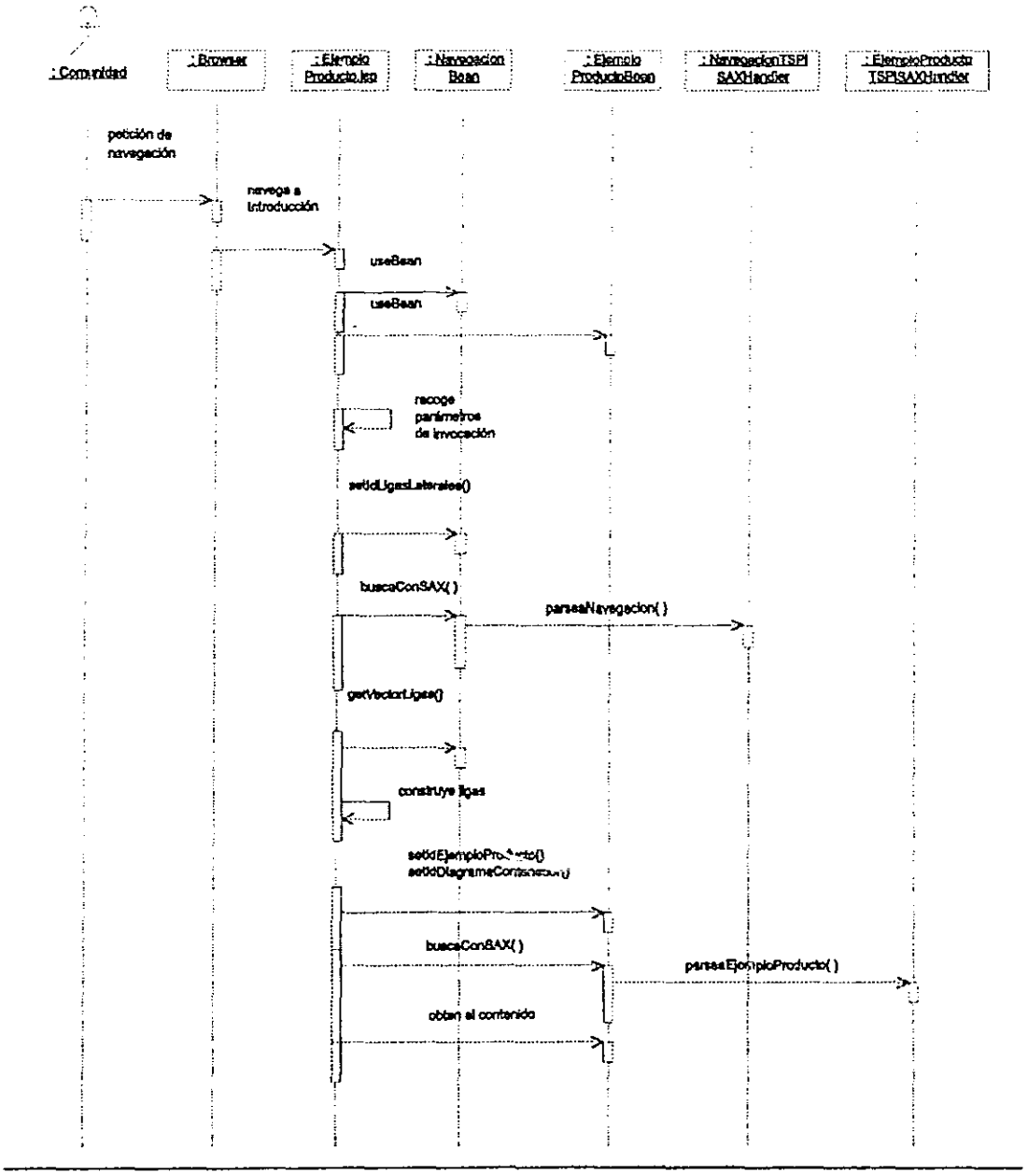


Figura. 4.9: Diagrama de Secuencia para F6.

CASO DE USO:	Acceder a Página de Guiones(para TSPi General, Ciclo 1 y n, Lanzamiento 1 y n, Estrategia 1 y n, Planeación 1 y n, Requerimientos 1 y n, Diseño 1 y n, Implementación 1 y n. Pruebas 1 y n, Postmortem 1 y n)
IDENTIFICADOR:	F7

DIAGRAMA DE SECUENCIA:

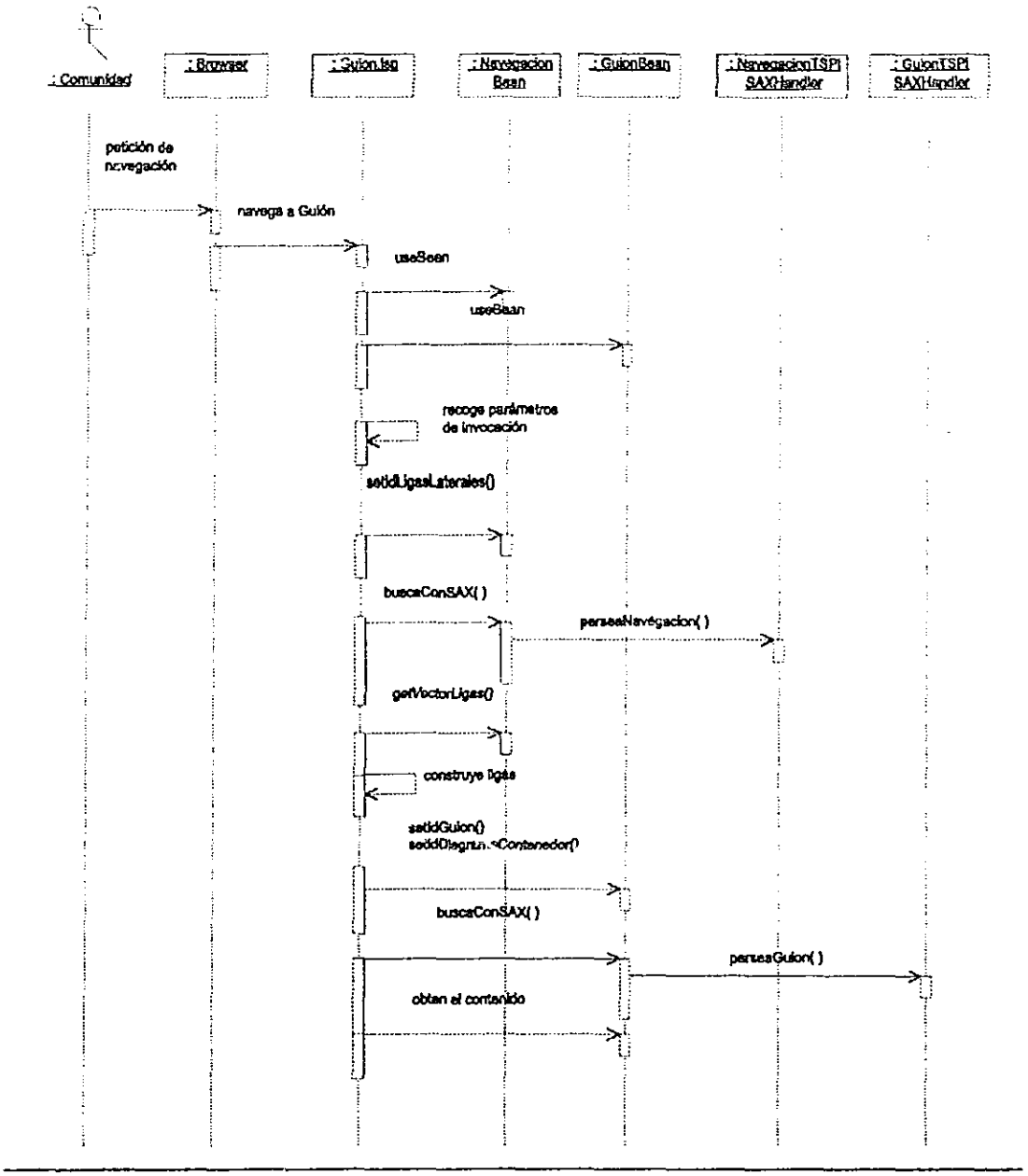


Figura. 4.10: Diagrama de Secuencia para F7.

4.3.5 Diseño XML

Elementos Generados

Se generaron archivos XML que contienen la información que necesita el sitio WEB TSPi para desplegarse.
A continuación se describen los elementos generados:

Se generaron los elementos `actividad` los cuales describen una actividad del sitio. Los elementos que contienen son:

`idactividad, iddiagramacontenedor, nombrediagramacontenedor, rutaactividad`

Se generaron los elementos `diagrama` los cuales describen un diagrama del sitio. Los elementos que contiene son:

`iddiagrama, nombrediagrama, rutadiagrama`

Se generaron los elementos `ejemploproducto` las cuales describen un ejemplo de algún producto contenido en el sitio. Los elementos que contiene son:

`idejemplo, iddiagramacontenedor, nombreejemplo, nombrediagramacontenedor, ejemplogif, numeropaginas, ancho640x480, alto640x480, ancho800x600, alto800x600, ancho1024x768, alto1024x768, rutaejemplo, forma`

Se generaron los elementos `guion` los cuales describen un guión del sitio. Los elementos que contiene son:

`idguion, iddiagramacontenedor, nombreguion, nombrediagramacontenedor, guiongif, numeropaginas, ancho640x480, alto640x480, ancho800x600, alto800x600, ancho1024x768, alto1024x768, rutaguion`

Se generaron los elementos `apartado` los cuales describen un apartado de la introducción del sitio. Los atributos de la tabla son:

`idintroducion, nombreintroduccion, introduccciongif, numeropaginas, ancho640x480, alto640x480, ancho800x600, alto800x600, ancho1024x768, alto1024x768, rutaintroduccion`

Se generaron los elementos `ligaslateral` las cuales describen los tipos de ligas laterales de la navegación de los diagramas, ejemplos, productos, guiones y actividades del sitio. Los atributos de la tabla son:

idligaslaterales, nombretspiligalateral, rutatspiligalateral, nombrecicloligalateral, rutacicloligalateral, nombrefaseligalateral, rutafaseligalateral, nombreextensionligalateral ,rutaextensionligalateral

Se generaron los elementos producto los cuales describen un guión del sitio. Los atributos de la tabla son:

idproducto, iddiagramacontenedor, nombrediagramacontenedor, rutaproducto

Parser XML

El parser elegido es **SAX**, esto es debido a su ahorro en cuestión de memoria.

4.4 Prototipo del Sitio

4.4.1 Formato de los Tipos de Pantallas

Los tipos de pantallas manejados son de *Bienvenida*, *Introducción*, *Diagrama*, *Actividad*, *Producto*, *Ejemplo de Producto*, y *Guión*.A continuación se muestra una descripcón del formato de cada tipo de pantalla:

Bienvenida Ésta contiene una sección de texto en HTML(1), una sección para una barra de menú principal para navegar en el sitio(2).

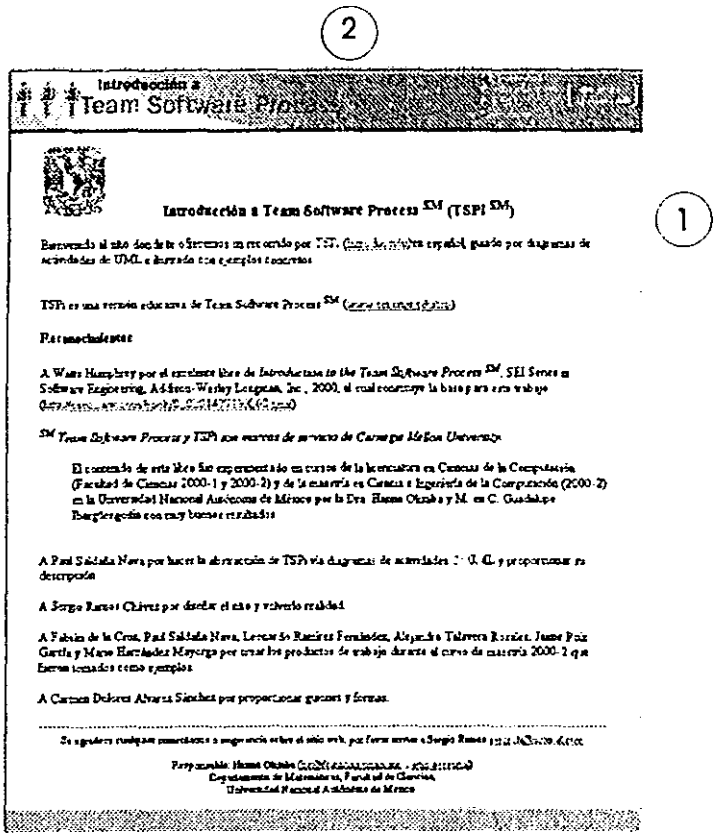


Figura. 4.11: Pantalla de Bienvenida.

Introducción Ésta contiene una sección dedicada a un texto en imagen(1), una sección para una barra de menú principal para navegar en el sitio(2). Se incluye una barra de menú secundario para navegar en la introducción(3). Éste tipo de pantalla cuenta con una sección para navegar a través de las páginas de introducción(4), además de una sección para bajar el documento desde la red(5).

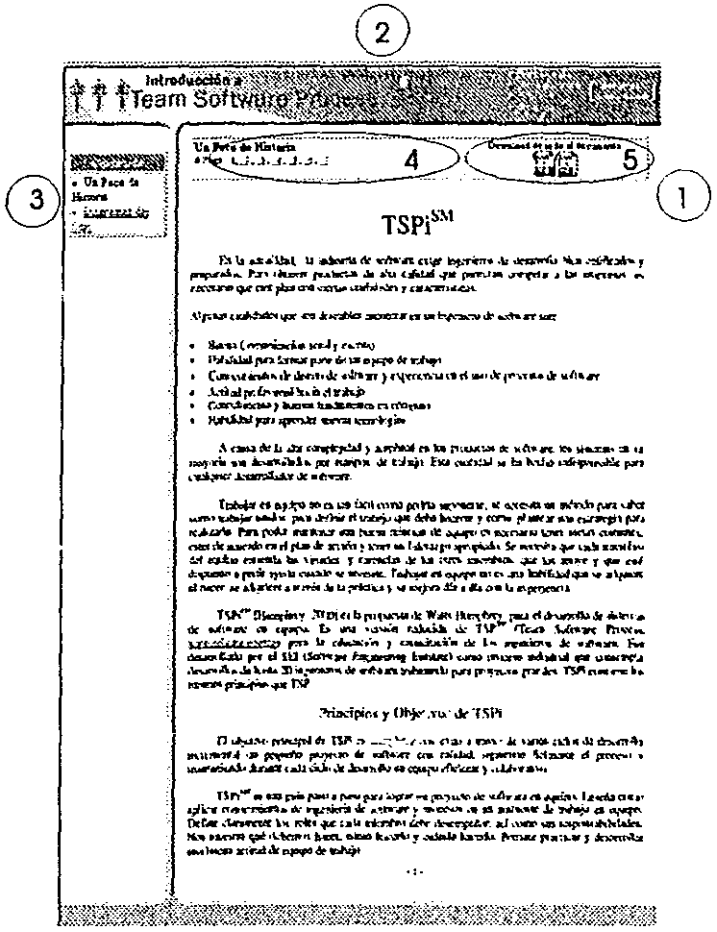


Figura. 4.12: Pantalla de Introducción.

Diagrama Ésta contiene una sección dedicada a un diagrama en imagen(1), una sección para una barra de menú principal para navegar en el sitio(2). Se incluye una barra de menú secundario para navegar en TSPi(3).

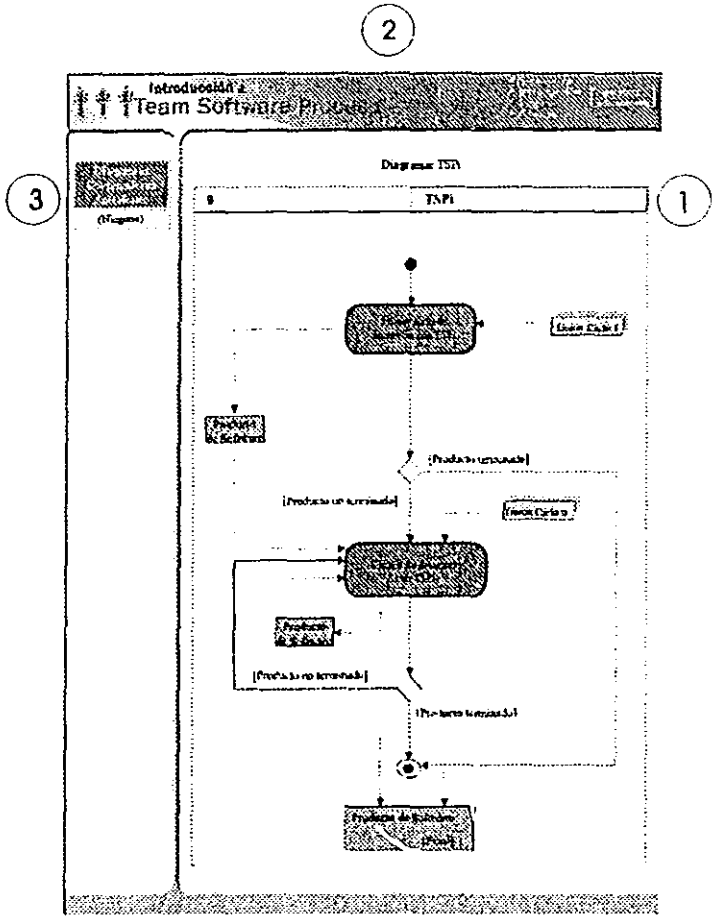


Figura. 4.13: Pantalla de Diagrama.

Actividad Ésta contiene una sección dedicada a la descripción de una actividad en texto HTML(1), una sección para una barra de menú principal para navegar en el sitio(2). Se incluye una barra de menú secundario para navegar en TSPi(3).

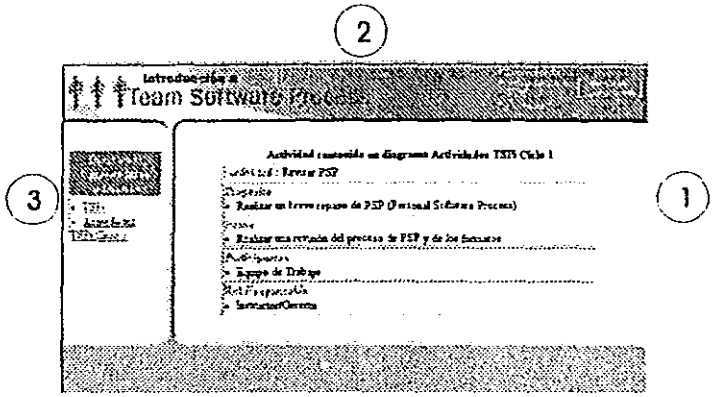


Figura. 4.14: Pantalla de Actividad.

Producto Ésta contiene una sección dedicada a una lista de productos en texto HTML(1), una sección para una barra de menú principal para navegar en el sitio(2). Se incluye una barra de menú secundario para navegar en TSPi(3).

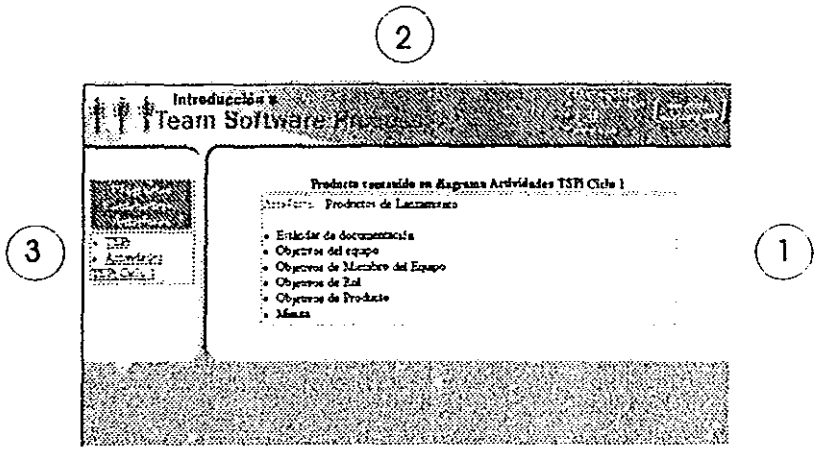


Figura. 4.15: Pantalla de Producto.

Ejemplo de Producto Ésta contiene una sección dedicada a un ejemplo de producto en imagen(1), una sección para una barra de menú principal para navegar en el sitio(2). Se incluye una barra de menú secundario para navegar en TSPi(3). Éste tipo de pantalla cuenta con una sección para navegar a través de las páginas del ejemplo(4), además de una sección para bajar el documento desde la red(5).

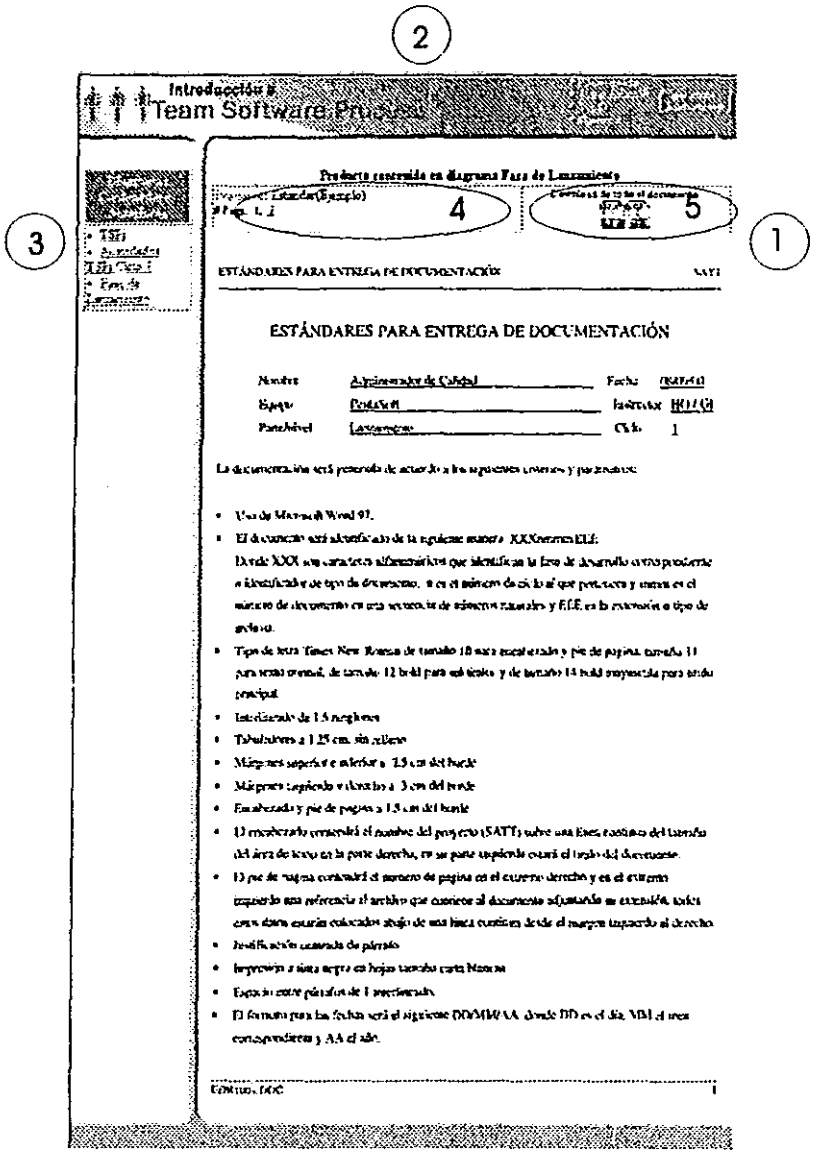


Figura. 4.16: Pantalla de Ejemplo de Producto.

Guión Ésta contiene una sección dedicada a un guión en imagen(1) de TSPi, una sección para una barra de menú principal para navegar en el sitio(2). Se incluye una barra de menú secundario para navegar en TSPi(3). Éste tipo de pantalla cuenta con una sección para navegar a través de las páginas del guión(4), además de una sección para bajar el documento desde la red(5).

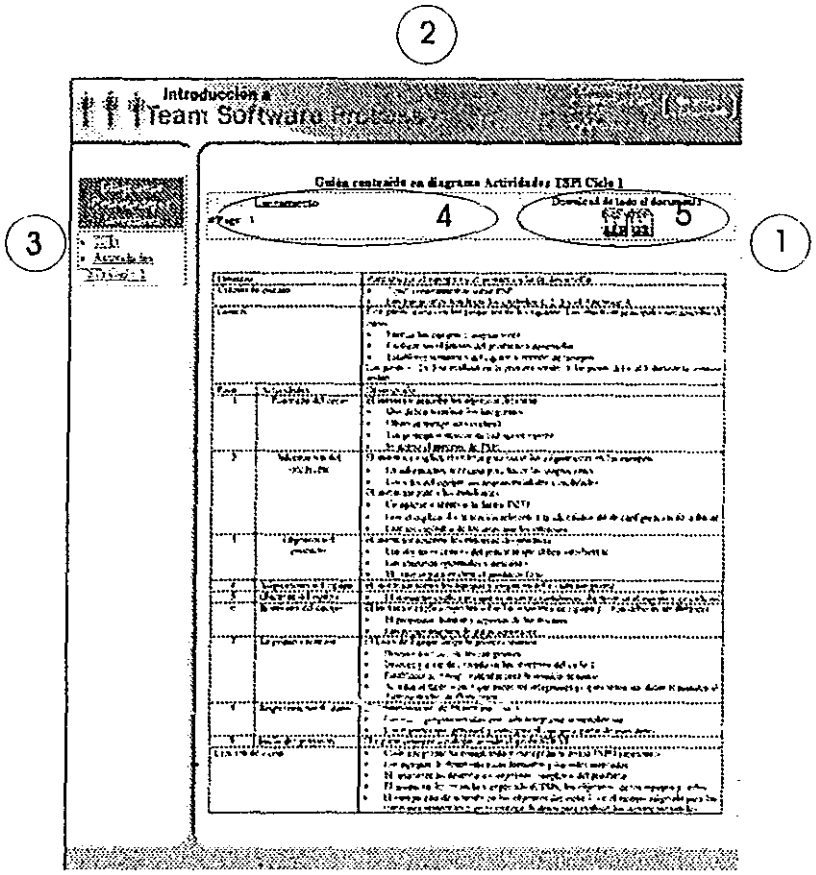


Figura. 4.17: Pantalla de Guión.

4.4.2 Arquitectura del Sitio

Para una visión global del sitio se presenta un árbol de navegación, en el que se muestran las pantallas y su dependencia con otras pantallas.

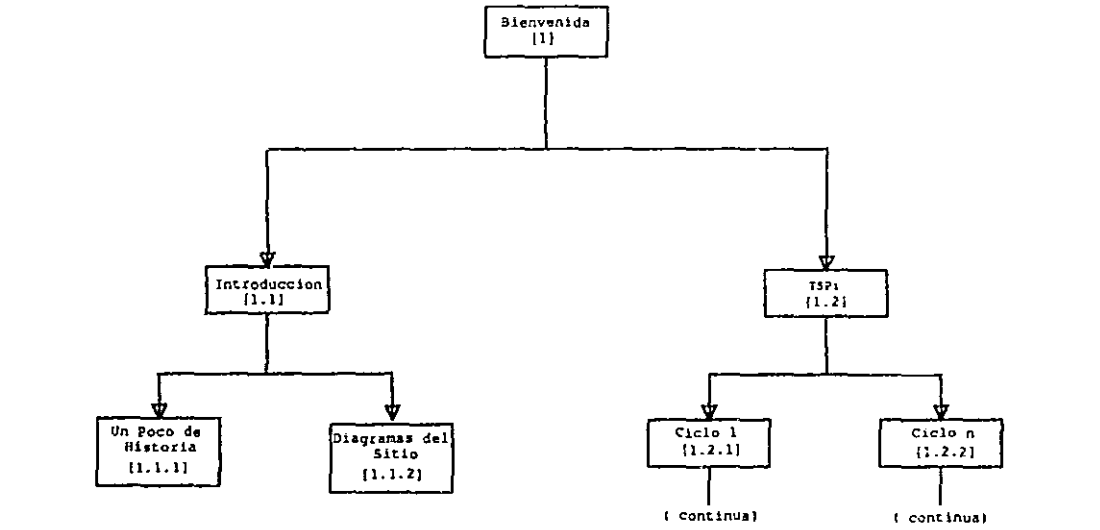


Figura. 4.18: Pantalla de Navegación 1.

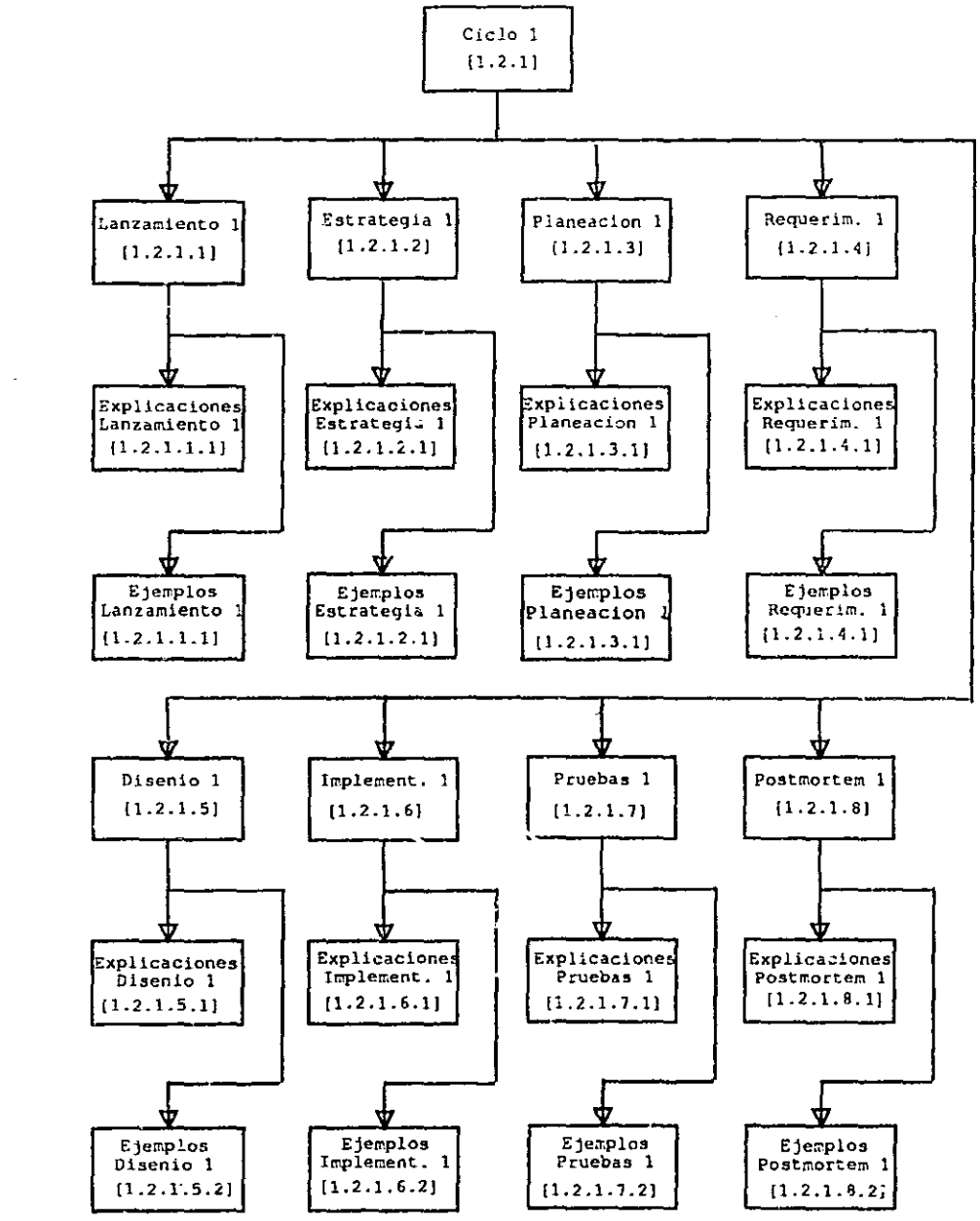


Figura. 4.19: Pantalla de Navegación 2.

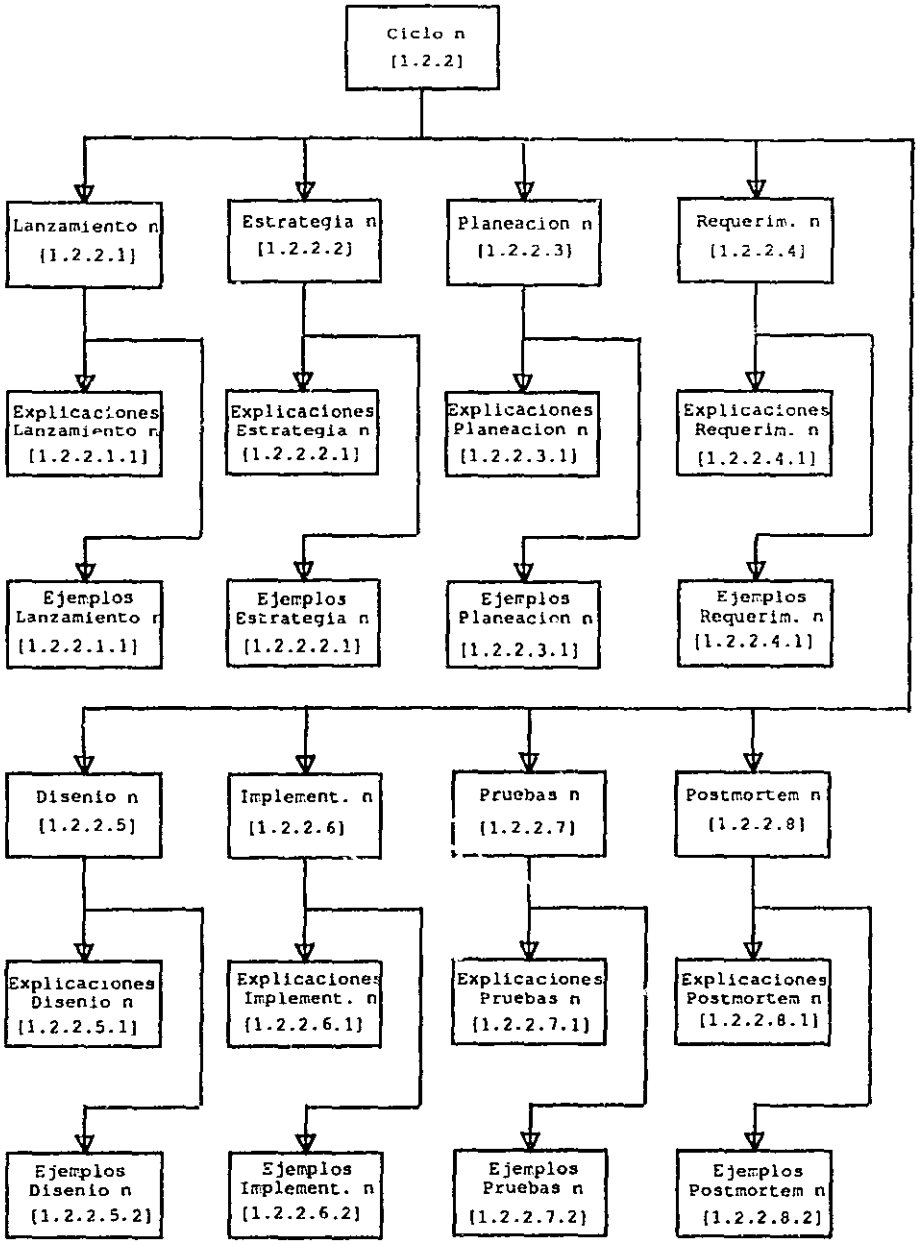


Figura. 4.20: Pantalla de Navegación 3.

4.5 Experiencias de Uso del Sitio

El sitio WEB ha sido usado ya en cursos de Ingeniería de Software en la Facultad de Ciencias de la UNAM. En particular, por primera vez fué empleado en el curso del semestre 2001-II. Al término del primer ciclo del curso mencionado, se realizó una encuesta con la finalidad de efectuar un monitoreo del apoyo y aceptación del sitio.

La encuesta se realizó con los 23 alumnos del curso de Ingeniería de Software. Básicamente en un curso que hace uso del sitio, se manejan tres fuentes de información las cuáles son las clases, el libro de TSPi, y el sitio. Tomando en cuenta éste aspecto se redactó el siguiente cuestionario:

---

1. ¿Qué tanto apoyo consideras que te dió el libro TSPi para tu desenvolvimiento en el ciclo 1 del curso?
   a) Ningún apoyo b) Poco apoyo
   c) Buen apoyo d) Muy Buen apoyo

2. ¿Qué tanto apoyo consideras que te dió el sitio WEB TSPi para tu desenvolvimiento en el ciclo 1 del curso?
   a) Ningún apoyo b) Poco apoyo
   c) Buen apoyo d) Muy Buen apoyo

3. ¿Qué tan claro consideras que es el sitio?
   a) Nada claro b) Poco claro
   c) Claro d) Muy Claro

4. Escribe el porcentaje de información que recibiste por parte de la clase, del libro, y del sitio para tu desenvolvimiento en el ciclo 1, de tal forma que la suma de los porcentajes sea 100%

% Clase	
% Libro	
% Sitio	
% Total	100.00

A continuación se presenta un resumen de las respuestas obtenidas.

Las respuestas obtenidas para la primer pregunta "¿Qué tanto apoyo consideras que te dió el libro TSPi para tu desenvolvimiento en el ciclo 1 del

curso?" fueron:

a) Ningún apoyo	0	b) Poco apoyo	8
c) Buen apoyo	15	d) Muy Buen apoyo	0

Las respuestas obtenidas para la segunda pregunta "¿Qué tanto apoyo consideras que te dió el sitio WEB TSPi para tu desenvolvimiento en el ciclo 1 del curso?" fueron:

a) Ningún apoyo	0	b) Poco apoyo	0
c) Buen apoyo	13	d) Muy Buen apoyo	10

Las respuestas obtenidas para la tercer pregunta "¿Qué tan claro consideras que es el sitio?" fueron:

a) Nada claro	0	b) Poco claro	2
c) Claro	12	d) Muy Claro	9

En promedio, las respuestas obtenidas para la cuarta pregunta "Escribe el porcentaje de información que recibiste por parte de la clase, del libro, y del sitio para tu desenvolvimiento en el ciclo 1, de tal forma que la suma de los porcentajes sea 100%" fueron:

% Clase	36.52
% Libro	24.78
% Sitio	38.70
% Total	100.00

A simple vista se puede observar, a partir de la cuarta pregunta, que el sitio aportó aproximadamente un tercio del apoyo para que los alumnos se desenvolvieran en el primer ciclo de desarrollo de TSPi(ver Figura 4.21). De cualquier forma, se pudieron efctuar algunas reflexiones sobre las respuestas de los alumnos, tomando como base su respuesta en la cuarta pregunta y observando el resto de sus respuestas.

De los resultados obtenidos, doce alumnos consideraron a la clase como una de las fuentes de información de mayor apoyo. De éstos alumnos, el 80% pensaron que el libro era un buen apoyo, y el 17% dijo que era de poca ayuda. En cuanto al apoyo que les brindo el sitio, el 58% pensó que el sitio había dado un buen apoyo, mientras que el 42% pensó que el sitio les había dado un muy buen apoyo. En lo que respecta a la claridad del sitio, el 17% pensó que el sitio era poco claro, el 41% pensó que el sitio era claro, y el 42% pensó que el sitio era muy claro.

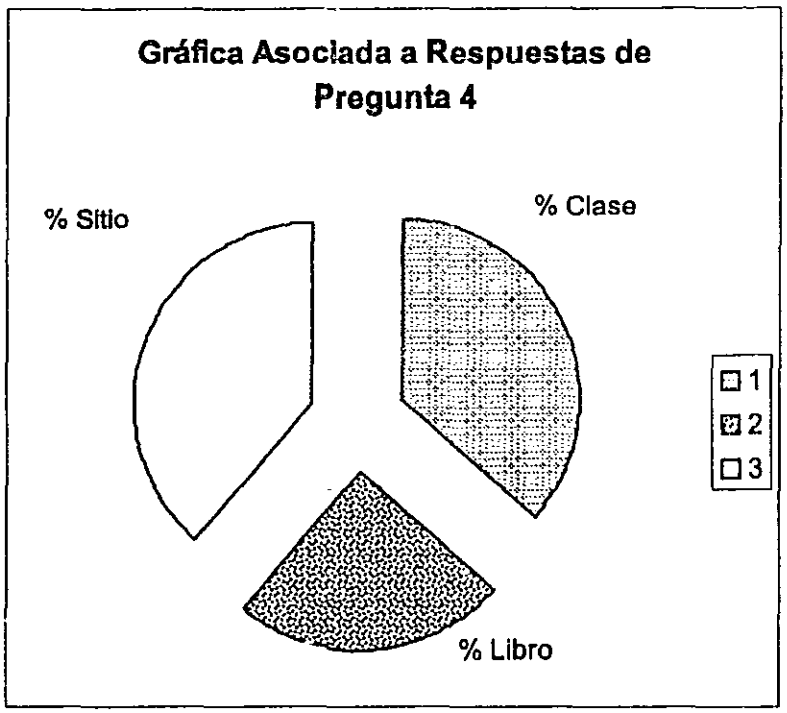


Figura. 4.21: Gráfica de Pregunta 4.

También de los resultados obtenidos, dos alumnos consideraron al libro de TSPi como una de las fuentes de información de mayor apoyo. Éstos alumnos, pensaron que el sitio es un buen apoyo, y que era claro.

Por último, de los resultados obtenidos, doce alumnos consideraron al sitio WEB como una de las fuentes de información de mayor apoyo. De éstos alumnos, el 58% pensó que el libro les brindaba poco apoyo, y el 42% dijo que era un buen apoyo. En lo que respecta a la claridad del sitio el 58% pensó que era claro, y el 42% pensó que era bastante claro.

Como se puede apreciar, el apoyo y aceptación del sitio WEB son buenos y lo más importante es que auxilian a los alumnos en el proceso de aprendizaje de la metodología. Es importante remarcar que el sitio no busca sustituir el libro *Introduction to the Team Software Process* de Watts S. Humphrey, sino únicamente servir de apoyo visual y práctico.

Capítulo 5

Conclusiones

El desarrollo de sitios WEB es una tendencia que actualmente está en aumento. Todo mundo quiere aprovechar la difusión que un sitio WEB puede lograr en Internet , pero éste como cualquier otro producto de software requiere seguir las normas de una metodología que le permitan obtener un alto grado de calidad.

La metodología de TSPi me permitió experimentar el impacto de la Ingeniería de Software en éste tipo de proyectos relativamente nuevos. Pude constatar las ventajas de un proyecto bien documentado y bien administrado, con uso de formas, estándares, y documentos descriptivos tanto de las necesidades de los clientes así como de las ideas de los programadores. Desde el punto de vista del cliente, ciertamente algunos de éstos documentos son de tal importancia que fungen incluso como contratos de lo que el cliente desea con lo que el programador entiende que el cliente desea. Desde el punto de vista del desarrollador una documentación del estilo de la de TSPi, le permite saber con precisión qué hacer, cómo hacerlo y cuándo hacerlo, además de que la metodología favorece un espíritu de colaboración en equipos de trabajo, que actualmente es la tendencia para la generación de software. La manera en la cuál usé ésta metodología me ayudo a conocer más a fondo todos los roles que intervienen en el desarrollo de los proyectos de software en equipo, pues al inicio del proyecto solo tenía experiencia como Administrador de Soporte.

En lo que respecta a los objetivos iniciales del proyecto, éstos fueron cubiertos. El sitio es empleado en los cursos de Ingeniería de Software de la Facultad de Ciencias de la UNAM para la enseñanza de TSPi. Además, el desarrollo del sitio WEB estuvo en todo momento bajo la guía de TSPi.

En el aspecto técnico, logré incrementar mi conocimiento de todas las herramientas involucradas en el desarrollo del sitio, las cuáles son tecnologías de punta para éste tipo de proyectos. A la vez, comprendí que no solo el

conocimiento obtenido durante la carrera es importante, sino que el constante estudio de nuevas tecnologías es fundamental para poder obtener la mejor decisión en cuestión de herramientas, y que sin duda estas también afectan la calidad del resultado final.

Apéndice A

Anexo del Capítulo 4

En el presente apéndice sólo se indica que la documentación completa de cada ciclo de desarrollo, se encuentra en el disco que acompaña a éste trabajo. A contunuación se mencionan los directorios que se encuentran en el disco y una indicación del contenido de los mismos:

- **ciclo de desarrollo 1:** contiene la documentación y productos generados para el ciclo 1
- **ciclo de desarrollo 2:** contiene la documentación y productos generados para el ciclo 2
- **ciclo de desarrollo 3:** contiene la documentación y productos generados para el ciclo 3
- **ciclo de desarrollo 4:** contiene la documentación y productos generados para el ciclo 4
- **codigo final:** contiene el código del último prototipo

Bibliografía

[1] Ford G., *A Progress Report on UnderGraduate Software Engineering Education*, Software Engineering Institute Carnegie Mellon University, 1994.

[2] Ghezzi Carlo, *Programming Language Concepts*, John Wiley & Sons, Tercera Edición, 1997.

[3] Von Maryhauser Annaliese, *Software Engineering: Methods and Management*, Academic Press Inc., 1990.

[4] Scacchi W., *Models of Software Evolution*, Software Engineering Institute Carnegie Mellon University, 1987.

[5] Grady, R.B. and D. I. Caswell, *Metrics: Establishing a Company-Wide Program*, Prentice Hall, 1987.

[6] Tripp L. Leonard, *Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society, 2001.

[7] Watts S. Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, 1995.

[8] Hayes W. and Over J., *The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers*, Software Engineering Institute Carnegie Mellon University, 2001.

[9] McAndrews R. Donald, *The Team Software Process^{SM} (TSP^{SM}): An Overview and Preliminary Results of Using Disciplined Practices*, Software Engineering Institute Carnegie Mellon University, 2000.

[10] Watts S. Humphrey, *Introduction to the Personal Software Process^{SM}*, Addison-Wesley, 1996.

[11] Watts S. Humphrey, *Introduction to Team Software Process^{SM}*, Addison-Wesley, 2000.

[12] Anaya Multimedia, *HTML Creación de Páginas WEB*, Anaya Multimedia, 1998.

[13] Goodman Danny, *JavaScript Bible 3^d Edition*, IDG Books Worldwide, 1998.

[14] Walsh Aaron, Fronckowaik W. John, *Java Bible*, IDG Books Worldwide, 1998.

[15] David Flanagan, *Java En Pocas Palabras*, McGrawHill, 1998.

[16] Silberchatz, Korth, Sudarshan, *Fundamentos de Bases de Datos*, McGrawHill, 1998.

[17] Date, Jaime Malpica*(Traducción)*, *Introducción a los Sistemas de Bases de Datos*, Addison-Wesley, 1986.

[18] Diccionario en Línea `http://Dictionary.com`